Optimization of Multi-standards Software Defined Radio Equipments: A Common Operators Approach

Sufi Tabassum Gul

N° d'ordre : 3976 **ANNÉE 2009**

UNIVERSITÉ DE RENNES 1

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Traitement du Signal et Télécommunications

Ecole doctorale : Matisse

présentée par

Sufi Tabassum GUL

Préparée à l'unité de recherche : SCEE-SUPELEC/IETR UMR 6164
Nom développé de l'unité : Institut d'Electronique et de Télécommunications de Rennes
Composante universitaire : S.P.M.

Optimization of Multi-standards Software Defined Radio Equipments: A Common Operators Approach

Thèse soutenue à Supélec-Rennes le 12 Novembre 2009

devant le jury composé de :

Olivier SENTIEYS
Président, Professeur, Université de Rennes-I, Rennes, FRANCE

Michel AUGUIN
Rapporteur, Professeur, Université de Nice – Sophia Antipolis, FRANCE

Arnd-Ragnar RHIEMEIER
Rapporteur, Dr. – Ingénieur , EADS Defence and Security, Ulm, GERMANY

Guido MASERA
Examinateur, Professeur, Politecnico di Torino, ITALY

Jacques PALICOT
Directeur de thèse, Professeur, SUPELEC/IETR, Rennes, FRANCE

Christophe MOY
Co-directeur de thèse, Professeur, SUPELEC/IETR, Rennes, FRANCE

Dedications

To my parents
for their prayers, unmatched love and endless support

To Maryam,
To Asma & M. Hassan.

Acknowledgments

Research is an exciting journey and I started this journey by stepping onto the unpaved path of research three years ago in Signal Communication and Embedded Electronics Lab (SCEE)/Supélec-Rennes campus, FRANCE. By taking small strides I was able to draw a map of this journey and I hope it can be useful for other travellers. I have been down many frustrating sidings, but I have also found new useful paths that may lead to new discoveries, which is very rewarding and the source of my motivation.

First of all, countless thanks to Almighty Allah, Whose innumerable blessings, acknowledged and unacknowledged, perceptible and imperceptible, bestow our lives from cradle to grave. I express my deep and sincere gratitude to Almighty Allah Who has given me the knowledge, will and courage to take up this work and has bestowed upon me His blessings.

I wish to convey my warmest thanks to my parents who have given me endless support, encouraged me in all my decisions and provided me the opportunity with their sacrifices to reach this far with my studies.

I want to express my indebtedness to all, especially my professors, right from very beginning of my educational career till now who guided me throughout my life. I would like to express my gratitude to my good natured and devoted director and co-director of thesis, Dr. Jacques Palicot and Dr. Christophe Moy for their support and for always believing in my capabilities, particularly when I was lost for a while. I am very grateful to them for all their help, encouraging words during the years and constructively criticizing my work.

I especially want to acknowledge three members of our research team who actively participated in the collaborative work presented in last part of this thesis. Firstly, Dr. Ali Al-Ghouwayel, a former Ph.D student in SCEE lab, with whom I spent several weeks working on a combined research. Secondly, I acknowledge Laurent ALAUS who is working in CEA Grenoble for his cooperation and help. Thirdly, I am thankful to Dr. R. P. Mahesh for always providing me the valuable inputs.

I gratefully acknowledge all my former and present colleagues at the SCEE for creating such a pleasant work environment and for helping me to solve many problems. I also acknowledge 5050 team for their cooperation and technical support.

I am very thankful to the reviewers and examiners of my thesis for spending their precious time for reading, evaluating the quality of work and giving me extremely encouraging re-

marks.

Some friends have had a special impact on me during these three years. I would like to thank Abdul Sattar, an excellent researcher who gladly discussed anything from signal processing to Star Trek episodes. Also Sajjad Hussain is thanked for being an invigorating discussion partner.

I would particularly like to express my deepest thanks and appreciation to my wife and children, for their love and patience. You have been with me the whole journey. Thank you for all support, without you this thesis would never have been completed.

Finally, I would like to acknowledge the financial support provided by Higher Education Commission of Pakistan right from the very beginning to the end of my Ph.D. studies.

Sufi Tabassum Gul
Cesson Sévigné, FRANCE
November, 2009.

Contents

Abbreviations

2G	2^{nd} Generation
3G	3^{rd} Generation
4G	4^{th} Generation
3GPP	3^{rd} Generation Partnership Project
3GPP LTE	3GPP Long Term Evolution
ACTS	Advanced Communications Technologies and Services
A/D	Analog to Digital Converter
ADC	Analog to Digital Converter
ADSL	Asymmetric Digital Subscriber Line
AFE	Analog Front End
AGC	Automatic Gain Control
AGU	Address Generating Unit
ALUT	Adaptive Lookup Table
ALM	Adaptive Logic Module
AMPS	Advanced Mobile Phone System
ASIC	Application Specific Integrated Circuit
ATSC	Advanced Television Systems Committee
B3G	Beyond 3G
BC	Buliding Cost
BER	Bit Error Rate
BPF	Band Pass Filter
CB	Communication Block
CC	Computational Cost
CCM	Custom Computing Machine
CDMA	Code Division Multiple Access
CEA	Commissariat à l'Energie Atomique
CF	Common Function
CHARIOT	Changeable Advanced Radio for Inter-Operable Telecommunications
CO	Common Operator
CR	Cognitive Radio
CRC	Cyclic Redundancy Check
CRCC	Communications Research Centre of Canada
D/A	Digital to Analog Converter
DAC	Digital to Analog Converter
DAB	Digital Audio Broadcasting
DAG	Directed Acyclic Graph
DC	Direct Current

DECT	Digital Enhanced Cordless Telecommunications
DFE	Digital Front End
DFT	Discrete Fourier Transform
DFTFB	Discrete Fourier Transform FB
DIF	Decimation In Frequency
DIT	Decimation In Time
DMB	Digital Multimedia Broadcasting
DMFFT	Dual Mode FFT
DMR	Digital Modular Radio
DRM	Digital Radio Mondiale
DSamp	Down Sampling
DSP	Digital Signal Processor
DVB-H	Digital Video Broadcasting-Handheld
DVB-T	Digital Video Broadcasting-Terrestrial
E^3	End-to-End Efficiency
E^2R	End-to-End Reconfigurability
EA	Evolutionary Algorithms
EMI	Electro Mechanical Interference
EP	Evolutionay Programming
ER-LFSR	Extended Reconfigurable LFSR
ES	Exhaustive Search
EVS	Evolution Strategies
FB	Filter Bank
FD	Frequency Domain
FER	Frame Error Rate
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FFT-$\mathbb{C}$	FFT defined over complex field
FFT-GF2	FFT defined over $GF(2^m)$
FIR	Finite Impulse Response
FLMS	Fast Least Mean Squares
FM3TR	Future Multi-band Multi-waveform Modular Tactical Radio
FNT	Fermat Number Transform
FPGA	Field Programmable Gate Array
FRM	Frequency Response Masking
GA	Genetic Algorithms
GCU	Global Control Unit
GF	Galois Field
GFB	Goertzel FB
GMSK	Gaussian Minimum Shift Keying
GPP	General Purpose Processor
GPS	Global Positioning System
GSM	Global System for Mobile Communication
HDTV	High Definition TV
HF	High Frequency
HLL	High Level Language

HR	Hardware Radio
IC	Integrated circuit
IF	Intermediate Frequency
IIR	Infinite Impulse Response
INFOSEC	INFOrmation SECurity
I/O	Input/Output
I/Q	In phase/ Quadrature phase (modulator, data)
IS	Interim Standard for US Code Division Multiple Access
IS-95/136	Interim Standard 95/136
ISR	Ideal Software Radio
IST	Information Society Technologies
ITS	Information Technology Services
ITU	International Telecommunications Union
JTRS	Joint Tactical Radio System
LFSR	Linear Feedback Shift Register
LNA	Low Noise Amplifier
LO	Local Oscillator
LRA	Layered Radio Architecture
LSB	Least Significant Bit
LUT	Look Up Table
MAC	Multiplier ACcumulator
MAP	Maximum A Posteriori probability
MBMMR	Multi Band Multi Mode Radio
MExE	Mobile Execution Environment
MILS	Multiple Independent Levels of Security
MIT	Massachusetts Institute of Technology
MLSE	Maximum Likelihood Sequence Estimation
MMITS	Modular Multifunction Information Transfer System
MOP	Multi Objective Problem
MPRB	Modulated Perfect Reconstruction Bank
MSPS	Million Samples Per Second
MRateFilt	Multi Rate Filtering
NoC	Number of Calls
NTT	Number Theoretic Tranform
OFDM	Orthogonal Frequency Division Multiplexing
OS	Operating System
PA	Pragmatic Approach
PE	Processing Element
PFT	Pipelined Frequency Transform
PHY	Physical Layer
π-DQPSK	π Differential QPSK
PLD	Programmable Logic Device
PMCS	Programmable Modular Communications Systems
PMR	Professional Mobile Radio
PN	Pseudo Noise
PSK	Phase Shift Keying

QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
R-LFSR	Reconfigurable LFSR
RF-LFSR	Reconfigurable Fibonacci LFSR
RG-LFSR	Reconfigurable Galois LFSR
RS	Reed-Solomon
RF	Radio Frequency
RNRT	Réseau National de Recherche en Télécommunications
SA	Simulated Annealing
SATCOM	SATellite COMmunications
SCA	Software Communication Architecture
SCU	Stage Control Unit
SDR	Software Defined Radio
SoC	System on Chip
SR	Software Radio
SCR	Software Controlled Radio
SFDR	Spurious Free Dynamic Range
SIMO	Single-Input Multi-Output
SRC	Sample Rate Converters
TA	Theoretical Apporach
TAJPSP	Tactical AntiJam Programmable Signal Processor
TPFT	Tunable PFT
TQMFB	Tree Structured Quadrature Mirror FB
TV	Television
UFLMS	Unconstrained Frequency-domain Least Mean Squares
UHF	Ultra High Frequency
U.S.	United States of America
UMTS	Universal Mobile Telecommunication System
USR	Ultimate Software Radio
UTRA-FDD	UMTS Terrestrial Radio Access-Frequency Division Duplexing
UTRA-TDD	UMTS Terrestrial Radio Access-Time Division Duplexing
VHF	Very High Frequency
VLIW	Very Long Instruction Word
VITURBO	A Reconfigurable Architecture For Viterbi And Turbo
WiFi	Wireless Fidelity (IEEE 802.11)
WITS	Wireless Information Transfer System
WiMAX	Worldwide Interoperability for Microwave Access, Inc. (IEEE 802.16)
WLAN	Wireless Local Area Network
WP	Work Package
xDSL	Digital Subscriber Line Technologies

Résumé en français

Introduction générale

Cette thèse propose de contribuer à la conception des équipements de radio logicielle multi-standards sous un angle particulier comparativement aux efforts de recherche effectués en général dans la communauté. En effet, elle ne propose pas d'étudier comment il faut adapter les outils de conception actuels aux nouveaux défis proposés par la radio logicielle (co-conception matérielle-logicielle, outils d'abstraction de haut-niveau, etc.), mais comment ré-organiser les traitements afin d'exploiter de nouveaux degrés de liberté dans la conception.

La conception actuelle des équipements multi-standards consiste en une juxtaposition de circuits dédiés à chacun des standards. Dans un contexte à moyen terme, où de plus en plus de standards de communications sans fil sont destinés à être intégrés dans un même terminal, une telle approche va bientôt atteindre ses limites, aussi bien en termes de coût, d'encombrement que de possibilité d'offrir des services combinés entre les standards. On parle de conception de type velcro car on débranche une application radio pour en mettre une nouvelle en marche. On comprend bien les limites en termes de flexibilité, qui ne peut être que de très gros grain (au niveau d'un standard). A l'opposé, une approche radio logicielle pure repose sur l'utilisation d'opérateurs très fins (ceux du ou des processeurs) donc ré-utilisables de manière très flexible. Mais cela n'est réalisable que dans peu de cas réels. N'y aurait-il pas entre ces deux extrêmes des niveaux intermédiaires permettant de combiner à la fois les avantages des deux ? Cela consiste à trouver une décomposition à différents niveaux de granularité des traitements afin d'identifier et d'exploiter des redondances appelées "opérateurs communs". Cela ne concerne pas seulement les traitements associés à la couche physique, mais aussi toutes les couches du modèle OSI présentes dans un même équipement. L'objectif est d'optimiser la conception suivant différents critères impliquant par exemple la flexibilité, la vitesse de reconfiguration ou la surface occupée sur une cible matérielle. Cette étude se positionne dans le contexte plus large de la paramétrisation. Le terme paramétrisation regroupe l'ensemble des techniques utilisées pour modifier une opération par un simple changement de paramètres (et non en modifiant sa structure de base).

Le manuscrit est découpé en trois parties. La partie I est constituée de deux chapitres. Le chapitre 1 rappelle ce qu'est la radio logicielle et les défis qui y sont associés en termes de conception. Le chapitre 2 porte sur les techniques de paramétrisation pour la radio

logicielle et en particulier l'approche "opérateurs communs".

La partie II comporte trois chapitres. Le chapitre 3 explique comment modéliser les standards de communications sans fil sous forme de graphes hiérarchiques. Le chapitre 4 évoque les problèmes des paramètres de coût et les techniques d'optimisation associées. Le chapitre 5 explique le choix de la technique d'optimisation multi-critères utilisée.

La partie III est composée de quatre chapitres, chacun traitant d'un cas concret de conception illustrant la méthode issue des travaux. L'utilisation de l'opérateur FFT double mode fait l'objet de chapitre 6, le chapitre 7 étudie différentes variantes de l'opérateur LFSR, dans le chapitre 8 les bancs de filtres à réponse masquée sont ciblés au départ, puis un exemple de conception à plus grande échelle est étudié pour conclure dans le chapitre 9.

Enfin, en dernier lieu, des conclusions sont données à ce travail ainsi que des perspectives pouvant étendre ses résultats à l'avenir.

Introduction à la partie I

La première partie de ce manuscrit positionne dans un premier chapitre les travaux dans le contexte de conception associé à la radio logicielle, et plus précisément à la conception d'équipements de radio logicielle multi-standards. En effet, la radio logicielle a très tôt été identifiée comme une solution prometteuse pour la conception de tels équipements puisque le principe premier est d'utiliser un même matériel pour effectuer différentes sortes de modulations/démodulations. Le chapitre 1 explique pourquoi cela est envisageable et comment la radio logicielle propose de le mettre en oeuvre. Il existe en fait plusieurs niveaux de réalisation, plus ou moins réalistes suivant les contraintes (technologiques ou de coût) et suivant l'échéance prévue pour que de tels équipements soient commercialisés. Un survol des projets marquants l'histoire de la radio logicielle permet de voir que l'élan est bien réel dans ce domaine de recherche. Enfin, de nombreux défis restent à relever en termes de conception radio logicielle et nous positionnerons l'approche proposée dans ce travail par rapport à celles proposées par ailleurs dans la communauté de la radio logicielle.

Dans le second chapitre de cette partie, les techniques de paramétrisation sont décrites, avec différents niveaux envisagés : fonctions communes et opérateurs communs. Deux exemples d'opérateurs communs sont détaillés, à titre d'illustration : FFT et LFSR.

1 Emergence de la radio logicielle

1.1 La radio logicielle

La radio logicielle est à l'origine un ensemble de techniques visant à répondre aux évolutions de la conception des équipements de radiocommunications. Elle est ensuite aussi la source de nouvelles utilisations possibles des équipements radio (en cours de fonctionnement), grâce aux propriétés de flexibilité apportées par la radio logicielle. Ce n'est pas le coeur du sujet des travaux présentés ici si ce n'est que la contrainte de flexibilité est considérée comme une priorité afin de permettre à l'équipement de modifier de fonctionnement en temps-réel.

Suivant l'architecture de l'émetteur-récepteur, on peut distinguer plusieurs déclinaisons plus ou moins généralisées du principe de la radio logicielle, depuis la structure superhétérodyne de la Fig. 1 (très peu "radio logicielle") jusqu'à celle de la radio logicielle idéale de la Fig. 2(a) en passant par celle plus réaliste (dans certains cas) de la Fig. 2(b) et enfin la version de la radio logicielle réalisable dans de nombreux cas, celle de la Fig. 3.

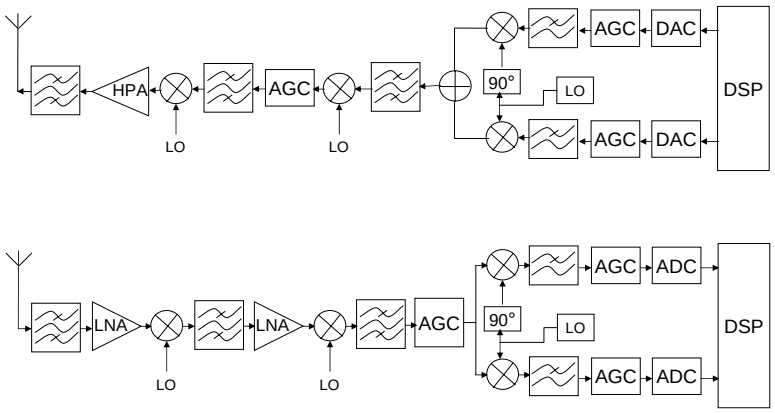

Figure 1: Emetteur/récepteur super-hétérodyne

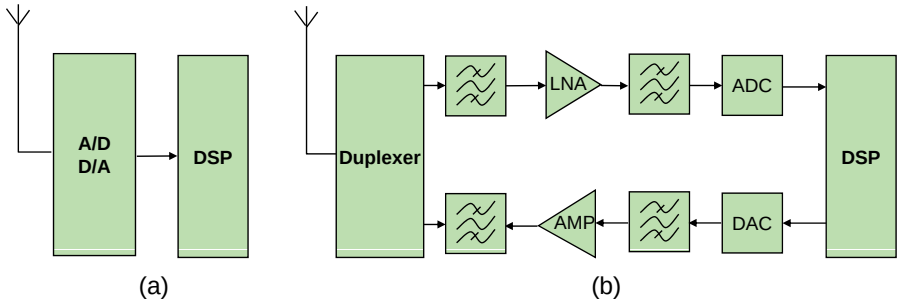

Figure 2: Une architecture réaliste d'une radio logicielle

Dans le cas de la radio logicielle, la conversion analogique numérique est directement effectuée en radio fréquence, suivie de processeurs pour effectuer les calculs du récepteur. Cela est souvent impossible en raison des contraintes technologiques que cela implique sur les convertisseurs et les processeurs. Ainsi, la radio logicielle restreinte (*Software Defined Radio* - SDR) relaxe les contraintes. Par exemple, elle peut correspondre à une numérisation en fréquence intermédiaire (cas de la Fig. 3) ou même en bande de base. On peut continuer à parler quand même dans ce cas de radio logicielle restreinte si la reconfigurabilité de l'équipement est exploitée. Dans le meilleur cas, une large bande de fréquence est alors directement numérisée englobant ainsi plusieurs signaux associés à différents standards. Cette fonctionnalité fait alors référence à un système dit multi-standards, capable d'opérer selon différents modes et offrant plusieurs services. Le passage d'un mode à un

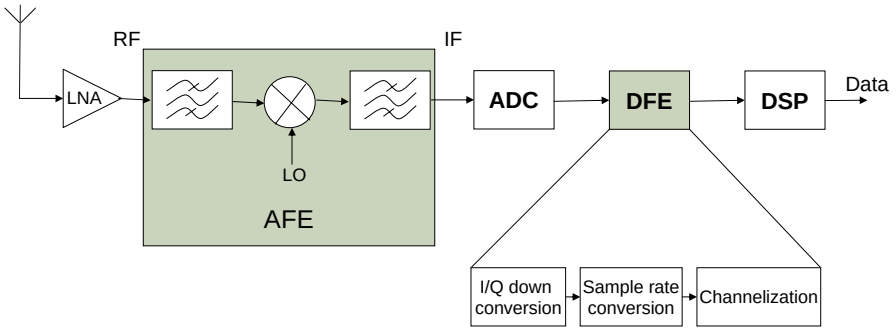

Figure 3: Une architecture de récepteur réalisable

autre est alors possible à condition que les processeurs de traitement du signal soient programmables ou reconfigurables.

Afin de réaliser d'une façon optimale un tel système, il est nécessaire de rechercher les points de convergence entre les standards à supporter par le terminal de façon à rendre les traitements communs. Cet aspect est connu sous le nom de la paramétrisation et est développé dans le chapitre suivant.

1.2 Les avantages de la radio logicielle

La radio logicielle est pour une grande partie une évolution naturelle de la radio car elle procure de nombreux avantages pour tous les acteurs du domaine, aussi bien les fabricants, les opérateurs (et fournisseurs de service) que les utilisateurs. La radio logicielle promet en effet dans ses principes des équipements multi-standards et multi-services. Il est possible d'en corriger les mal-fonctions, voire de les faire évoluer au niveau du traitement radio par téléchargement de logiciel, ce qui est à la fois bénéfique pour les utilisateurs (vous et moi, ainsi que les opérateurs), les fournisseurs de services et les constructeurs. Le rêve d'une mobilité totale peut être atteint grâce à la radio logicielle car elle permettra à l'équipement de s'adapter à tout standard en fonction des besoins. L'abandon des approches de conception de type velcro permet en outre aux fabricants d'envisager des économies puisqu'une même plate-forme peut supporter plusieurs standards même si ce n'est qu'avec un grand nombre de standards supportés que cela devient réellement le cas. En tout état de cause, le passage dans le domaine numérique d'une grande partie des traitements effectués en analogique auparavant permet de tirer bénéfice des techniques de conception numérique et de s'affranchir de beaucoup d'inconvénients liés à la conception analogique.

1.3 Projets et travaux les plus marquants de la radio logicielle

Historiquement, le projet *SPEAKeasy* a joué le rôle de projet pionnier dans le domaine de la radio logicielle, dès le début des années 90 [1, 2]. Il s'agit d'un projet mené par l'armée américaine dans le but de créer un modem reprogrammable qui devait supporter plusieurs

modes de transmission à plusieurs fréquences porteuses (de 2 MHz à 2 GHz). C'est de cette mouvance que sont apparus les experts américains qui ont crée le *SDR Forum* et les compagnies américaines leader du domaine.

Le projet JTRS (*Joint Tactical Radio System*) s'est attelé à définir et proposer une architecture logicielle afin d'abstraire la forme d'onde logicielle (application radio) de la plate-forme d'exécution matérielle [3, 4]. Le résultat obtenu est le SCA (Software Communication Architecture) [5] qui est une architecture ouverte ayant pour but de permettre à tout fournisseur de pouvoir proposer du matériel ou du logiciel compatible avec les sous-systèmes d'autres fournisseurs. Cela repose sur la définition d'interfaces (API - *Application Programming Interfaces*) en langage IDL (*Interface Definition Language*), l'usage du bus logiciel CORBA et du système d'exploitation POSIX. Le SCA est désormais un standard imposé pour tout fournisseur de l'armée américaine, mais il peine à faire preuve de son efficacité en termes de contraintes temps-réel.

Pendant ce temps, les universitaires américains se sont lancés également dans le domaine de la radio logicielle. Citons *Chariot* de Virginia Tech [2] et *SpectrumWare* du MIT. *SpectrumWare* en particulier chercha à évaluer la faisabilité d'utiliser un processeur général (GPP - General Purpose Processeur) pour mettre en oeuvre la radio logicielle, notamment pour des standards civils cette fois, comme AMPS et GSM [6].

Comme la radio se tourne vers le monde du logiciel, c'est tout naturellement qu'a émergé l'idée d'une GNU radio [7] dans le but d'ouvrir à une large communauté un outillage associant logiciel et matériel.

L'organe de référence au niveau international pour la radio logicielle est le *SDR Forum*. Il représente des sociétés, des universités et des organismes membres qui portent un intérêt pour le développement de la radio logicielle. C'est notamment le *SDR Forum* (secondé par l'*Object Management Group* - OMG - à un moment) qui encadre les développements réalisés autour du SCA.

L'Europe n'est pas en reste et la recherche sur le vieux continent est notamment soutenue par la Commission Européenne et les organismes nationaux d'aide à la recherche. De très nombreux projets sont liés au domaine de la radio logicielle (voir l'annexe A du document en anglais). A titre d'exemple le plus significatif, le projet E^2R (*End-to-End Reconfigurability*) et sa suite E^2R-II a regroupé plus de 30 partenaires pendant un total de quatre années dédiées à l'étude de la radio logicielle sous tous ses aspects.

1.4 Les défis de la conception radio logicielle

Comme déjà évoqué, la radio logicielle offre de nombreux avantages par rapport aux radios classiques, notamment grâce à la flexibilité qu'elle apporte. Cependant, concevoir un équipement de radio logicielle qui n'a plus un unique mode de fonctionnement, dont les paramètres peuvent varier énormément pendant le cycle de vie, qui associe à la fois du matériel et du logiciel, présente d'autres défis que ceux auxquels la conception d'un

équipement radio classique devait faire face.

En effet, il est nécessaire de faire appel à des techniques de co-conception logicielle/matérielle puisqu'un système de radio logicielle est désormais constitué d'unités de traitement qui exécutent une application radio en logiciel. Et cela serait trop simple ainsi. Pour des raisons d'efficacité en terme de traitement ou de consommation de puissance, il est en effet souvent obligatoire de combiner des unités de traitement de natures différentes, tels que les GPP, DSP (Digital Signal Processor), les FPGA (Field Programmable Gate Array) et les ASIC (Application-Specific Integrated Circuit) numériques et même analogiques. Cette pluralité et cette mixité rendent la conception très difficile. Il n'existe pas actuellement de solution intégrée qui permettent de le faire véritablement. Seules des solutions partielles existent. Cette étude mérite en soi plusieurs travaux de thèses. Citons une approche qui est utilisée dans l'industrie de la radio logicielle et qui repose sur Simulink. Cette approche permet de passer d'une simulation fonctionnelle (de type Matlab) à une implantation sur cible hétérogène associant des DSP, des FPGA et des convertisseurs grâce à certains artifices qu'il serait trop long de détailler ici. Citons également une approche de prototypage rapide organisée autour de SynDEx [8] qui permet de mettre en oeuvre rapidement une application radio logicielle sur différentes plates-formes matérielles.

Une solution prometteuse au niveau industriel, mais qui n'en est qu'à ses débuts et reste un espace de recherches, concerne l'utilisation d'approches de conception de haut niveau pour la radio logicielle. Comme en effet un système de radio logicielle est devenu un système complexe associant conception logicielle et matérielle, architecture de gestion et intergiciels (middleware), de nombreux modes de fonctionnement sont possibles. Une étape d'exploration de ces modes et d'exploration architecturale est nécessaire très en amont dans la phase de conception. C'est-à-dire qu'il est primordial de faire des choix bien avant l'implantation sur cible car d'une part celle-ci est très longue à obtenir, et d'autre part toute remise en cause si tard dans le cycle est très pénalisante en termes de coûts et de temps de développement. L'utilisation d'approches de conception de style MDA (*Model Driven Architecture*) qui repose sur une modélisation de type UML est particulièrement visée [9]. Le but est d'utiliser un flot basé sur UML tout au long du cycle de conception. Cela peut contribuer notamment à passer d'un niveau de modélisation à un autre par transformation de modèles [10] et peut aider à la production de documentation, tache primordiale dans une processus industriel certifié ou non.

Un autre effort particulièrement poussé dans le domaine de la radio logicielle concerne le développement d'outils permettant d'intégrer le SCA dans les équipements [11, 12, 13]. Ceci vise plus particulièrement les systèmes destinés à fonctionner dans un cadre militaire, mais on peut imaginer que le SCA devienne un standard même pour des applications civiles. L'approche proposée dans ce manuscrit n'est pas de cet ordre et peut tout aussi bien être utilisée dans un cadre SCA ou sans SCA.

2 Techniques de paramétrisation pour la radio logicielle

2.1 La paramétrisation

Le chapitre 2 du manuscrit expose les principes de la paramétrisation. Le but est de rechercher dans un premier temps les caractères communs entre les traitements de différents standards [14] puis d'en proposer des architectures communes et flexibles, à l'opposé d'une approche de type velcro. Un exemple est fourni dans la Fig. 4.

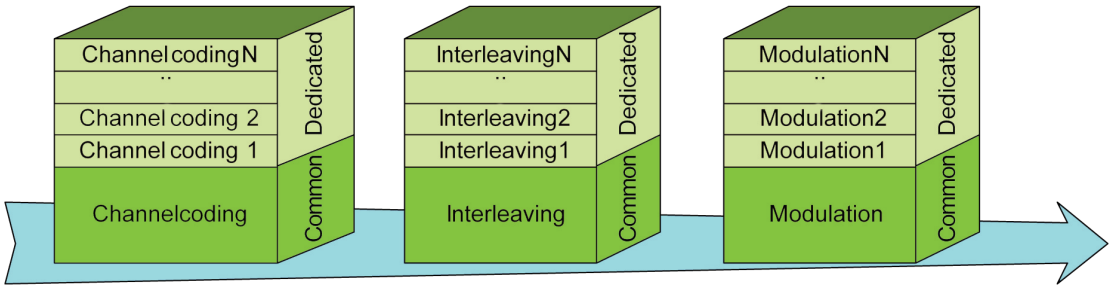

Figure 4: Taches d'un émetteur multi-standards

La paramétrisation se décline selon deux approches : l'approche théorique et l'approche pragmatique. L'approche théorique consiste à lister de façon hiérarchique tous les appels de fonctions possibles dans un terminal. A titre d'exemple, si l'on considère la modulation OFDM à gros grain, elle peut faire appel à la FFT à grain moyen, qui elle même peut faire appel à l'opérateur papillon et ce dernier faisant appel aux opérateurs arithmétiques classiques à grain très fin. On peut même envisager d'aller plus bas et retomber ainsi sur les portes logiques de base, voire les transistors. Mais il n'est sans doute pas nécessaire de descendre si bas. De cette façon, on pressent qu'il est possible de choisir un jeu d'opérateurs pertinents à un niveau de granularité adéquat pour la conception d'un équipement multi-standards donné.

L'autre approche, dite pragmatique, consiste dans un premier temps à identifier les traitements communs puis dans un second temps à réaliser un opérateur générique qui devra alors être reconfigurable par changement de paramètres.

Ces traitements communs peuvent être classés en deux catégories que nous proposons de détailler ci-après :

- fonctions communes,
- opérateurs communs.

2.2 Approche par fonctions communes

Le premier type de paramétrisation se fait au niveau fonction. Le niveau de granularité est élevé. Bien souvent, les travaux ayant eu pour objet de proposer des fonctions

communes aboutissaient à des opérateurs plus génériques que reconfigurables. Les structures proposées telles que [15] [16] sont certes communes mais prohibitives en terme de surface et surtout n'exploitent pas vraiment le principe de reconfiguration. Elles sont de plus peu évolutives car reliées à des standards prédéfinis. Cet aspect implique de diminuer le niveau de granularité. C'est l'approche opérateur commun comme illustré dans la Fig. 5.

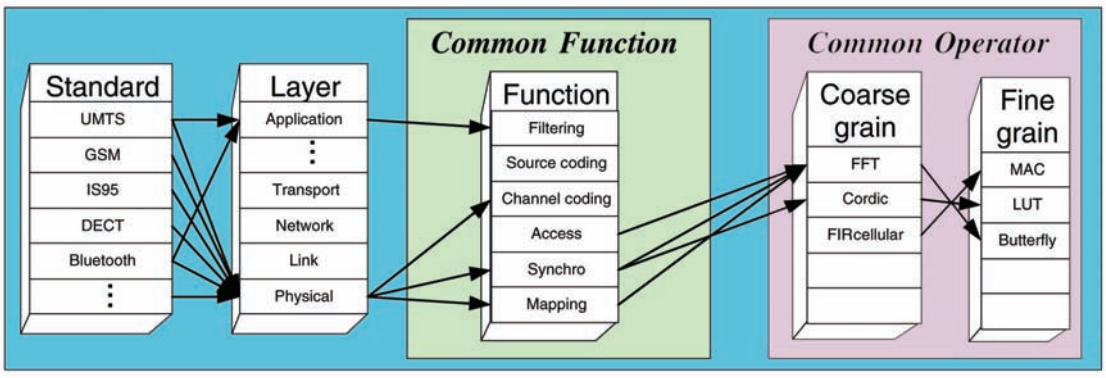

Figure 5: Les deux approches de paramétrisation, par fonction ou opérateur communs

2.3 Approche par opérateurs communs

La granularité est ici plus faible que celle des fonctions communes. L'idée est de tirer parti d'une structure de calcul existante et de la faire évoluer de façon à la rendre commune et reconfigurable. La réutilisation devient alors plus performante. L'étude présentée dans [17] peut être considérée comme pionnière sur le sujet des opérateurs communs. Elle a été ensuite poussée dans [18] et [19]. L'étude [17] prend l'exemple de l'opérateur FFT avec une approche pragmatique et constate que bon nombre de traitements tels que la (dé)modulation OFDM, l'égalisation, le filtrage, etc., peuvent être effectués avec l'opérateur FFT. L'objectif de cette thèse est d'apporter un cadre formel qui permet de justifier de telles affirmations et de les généraliser à tout autre opérateur pressenti ou non, existant actuellement ou non. Il est à remarquer que c'est un autre travail que d'identifier des opérateurs, comme cela est le cas des travaux d'autres chercheurs [18, 19].

D'après ces principes, on comprend qu'un opérateur commun est appelé plusieurs fois par différentes fonctions de niveau hiérarchique supérieur tout au long de l'exécution des opérations de traitement du signal effectuées par l'équipement. La ré-utilisation est une notion importante. Si l'on considère la décomposition proposée dans la Fig. 6 proposée dans [20], on constate que la chaîne de communication des différents standards qui doivent être implantés dans l'équipement est formée de différents blocs de traitement : codage de source, cryptage, codage de canal, modulation, etc. La fonctionnalité de chacun de ces blocs de traitement peut être également effectuée à partir de d'un ou plusieurs blocs d'un niveau inférieur, et ainsi de suite. Du haut en bas du graphe, la granularité est de plus en

plus fine.

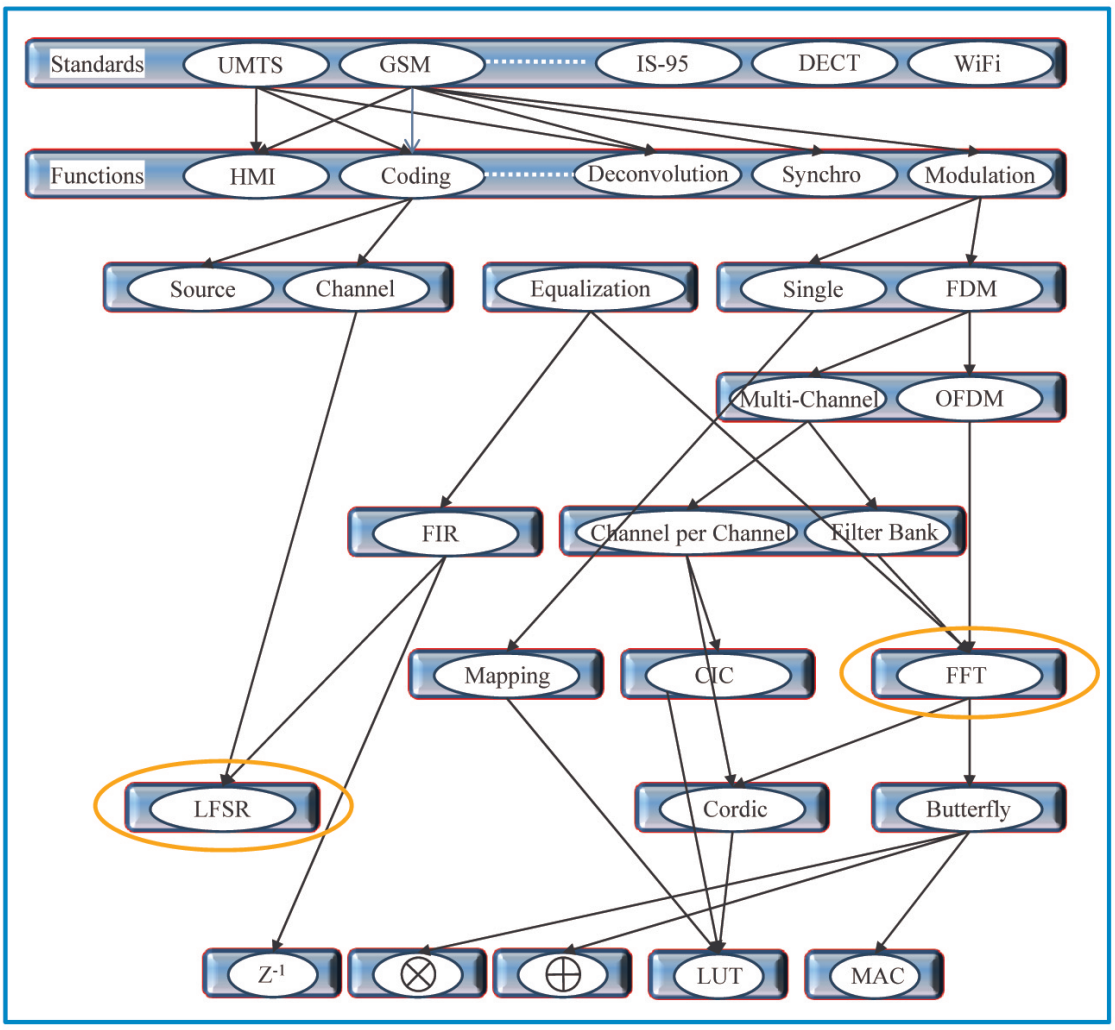

Figure 6: Diagrame généralisé de la décomposition de plusieurs standards

L'idée est de trouver le jeu d'éléments communs et de les partager entre les fonctionnalités de plusieurs taches de traitement. L'optimisation qui en découle est donc directement fonction de la manière dont sont exécutés ces opérateurs et de leur fabrication tant au niveau matériel que logiciel. Comme le but est d'atteindre le meilleur compromis entre performance et complexité, il est important de déterminer le niveau de granularité idéal que le concepteur devra choisir pour concevoir un équipement multi-standards, ni trop proche de l'approche velcro qui sera de grande complexité (mais hautement parallèle), ni trop proche des opérateurs arithmétiques qui sera très séquentielle (mais de complexité moindre).

2.4 Exemple d'opérateur commun

La Fig. 7 illustre comment la fusion de deux conceptions indépendantes (celles des deux côtés) peut influer sur les choix de conception. Les deux cas pris indépendamment ont visiblement deux alternatives de conception qu'il n'est pas évident de départager, alors que la mise en commun des deux cas semble nettement plaider en faveur du choix effectué dans la partie centrale de la figure. En effet, des motifs communs sont identifiables et il peut sembler judicieux d'en tirer partie.

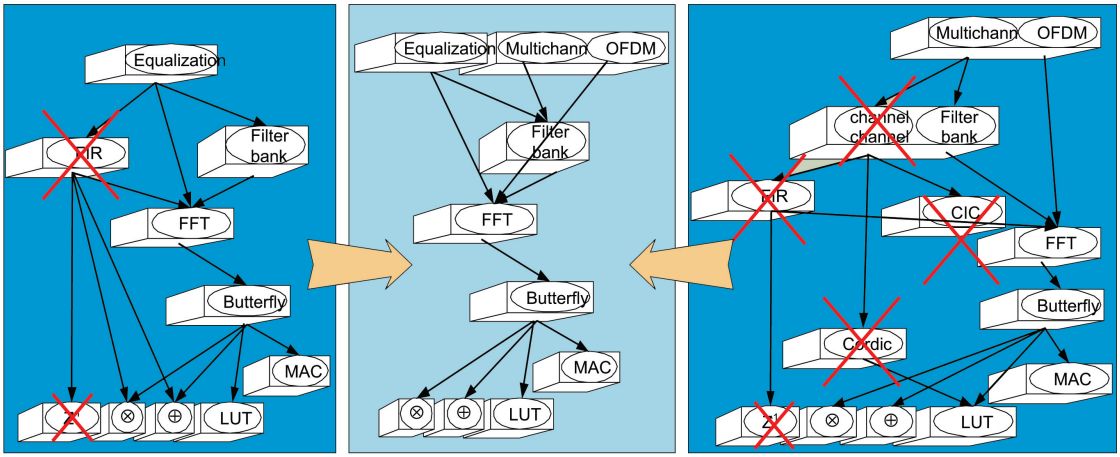

Figure 7: Chemin optimal pour la "channelization", l'égalisation et la démodulation OFDM

Introduction à la partie II

Dans cette partie, nous expliquons comment nous envisageons la modélisation d'un équipement multi-standards sous la forme d'un graphe. Mais il reste alors différentes alternatives qu'il faut convertir ensuite sous la forme d'un problème d'optimisation. Des coûts sont associés aux noeuds et aux arcs du graphe. On constate alors que différents coûts sont nécessaires, en fonction de la cible d'exécution matérielle (DSP, FPGA, etc.). Nous proposons ensuite une formulation de la fonction objectif associée au problème d'optimisation. Enfin plusieurs manières de résoudre le problème d'optimisation sont présentées.

Le chapitre 3 présente la structure de l'hypergraphe qui a été choisi pour poser le problème d'optimisation que nous cherchons à résoudre. Le détail des coûts utilisés est donné dans le chapitre 4, ainsi que la formulation de la fonction objectif à optimiser. Le chapitre 5 enfin concerne la méthode d'optimisation multi-critères elle-même en montrant sur un exemple simple comment elle fonctionne.

3 Modélisation graphique pour les systèmes de radio logicielle

3.1 Les graphes

Nous choisissons dans ce chapitre 3 le formalisme graphique convenant au problème que l'on cherche à résoudre. Survolons pour cela quelques principes et théories associées aux graphes. Les graphes sont utilisés dans un très grand nombre d'applications. Tout problème mathématique impliquant des points et des connexions peut être représenté sous la forme d'un graphe. Un graphe peut être représenté dans un plan ou dans un espace à trois dimensions par des sommets (ou noeuds) et des arêtes [21]. Sur la Fig. 8, les noeuds sont $\{u, v, w, x\}$ et les arêtes $\{a, b, c, d, e, f\}$. Le couple $\{a, b\}$ est une arête multiple car deux arêtes relient la même paire de points.

Catégories de graphes

Graphe simple : C'est un graphe qui n'a ni boucle ni arêtes multiples (dédoublées entre deux noeuds). Beaucoup de problèmes à base de graphes peuvent être ramenés à un graphe simple.

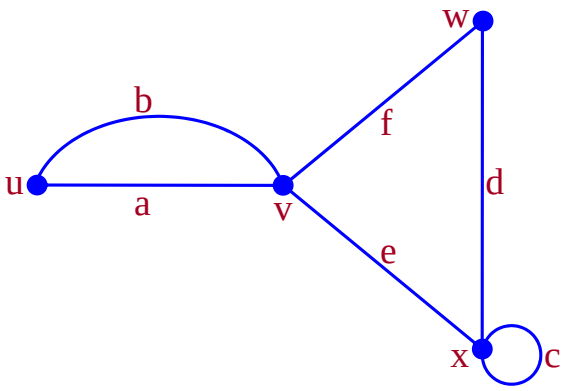

Figure 8: Exemple de graphe

Graphe orienté : C'est un graphe dont toutes les connexions sont unidirectionnelles. Une arête orientée est appelée arc. C'est un cas que l'on retrouve très souvent en ingénierie et dans le domaine scientifique. Des attributs peuvent être associés aux noeuds et aux arcs, tels que des poids, des coûts, etc.

Graphe acyclique orienté : C'est un graphe orienté qui n'a pas de cycle direct, c'est-à-dire de chemin fermé. Il s'agit de DAG en anglais (Direct Acyclic Graph). Même si cela semble être une contrainte forte, il existe de nombreux cas où les graphes acycliques s'appliquent, dès que les noeuds sont naturellement ordonnés (par exemple dans le temps ou en termes de hiérarchie).

Hypergraphe : Un hypergraphe est une généralisation d'un graphe dans laquelle une arête peut se connecter à plusieurs noeuds.

Travaux d'optimisation à partir de graphes

La théorie des graphes a été utilisée pour résoudre de nombreux problèmes d'optimisation, comme celui du voyageur de commerce [22] qui cherche à optimiser la distance à parcourir pour visiter tous ses clients. On en trouve une première forme en 1831 puis des travaux célèbres par Dantzig, Fulkerson et Jonhson [23] en 1954 puis Padberg et Rinaldi dans les années 80 [24]. Il y a plusieurs algorithmes pour trouver les chemins le plus court à travers un réseau, mais citons celui de Dijkstra [25] en 1959. Dans les années 60, les problèmes ont été classés par complexité. Edmonds a étudié les problèmes qui peuvent être résolus en un temps polynomial [26]. Cook [27] et Karp [28] ont établi les principes N-P complets.

Théorie des graphes et conception SDR

La théorie de graphes a déjà été utilisée dans le contexte de la conception d'équipements de radio logicielle. C'est le cas de l'approche utilisée pour le prototypage rapide dans [8]. La méthode de conception s'appuie sur l'utilisation de l'outil SynDEx [29] de l'INRIA qui repose sur la théorie des graphes pour déployer un graphe d'application sur un graphe

d'architecture matérielle. SynDEx propose un partitionnement ordonnancement optimisé pour le déploiement d'une application logicielle sur une plate-forme matérielle multi-processeurs. Dans le graphe d'application, les noeuds sont des traitements et les arcs représentent des échanges de données. Dans le graphe d'architecture matérielle, les noeuds sont des unités de traitement (typiquement des processeurs) et les arêtes des médias de communication. A chaque noeuds du graphe d'architecture correspond un temps d'exécution pour chaque traitement du graphe d'application que l'unité de traitement supporte. A chaque type de données de l'application correspond un temps de transfert sur le média du graphe d'architecture. L'objectif de l'optimisation de SynDEx est de minimiser le temps global d'exécution à partir des temps d'exécution des traitements sur chaque processeur et des temps de communication. SynDEx utilise pour cela un algorithme glouton. Il est à noter que SynDEx est originellement dédié au monde des stations de travail reliées en réseau. L'effort effectué dans les travaux [30] et [8] a permis de rapprocher la méthodologie vers le monde de l'embarqué. Cela s'est effectué en particulier pour des besoins en termes de conception radio logicielle et d'applications vidéo.

Le travail proposé dans cette thèse, bien que portant également sur la conception radio logicielle et reposant sur la théorie des graphes, est cependant très différent et attaque le problème de manière orthogonale. En effet, ce n'est pas un graphe flot de données qui est envisagé ici, mais un graphe structurel. Les liens entre les noeuds des Fig. 6 et 7 ne représentent pas des échanges de données, mais des relations de hiérarchie structurelle. L'ordonnancement n'est pas pris en compte dans les travaux présentés pour cette thèse.

Discussion

Dans le cas des graphes acycliques orientés ou DAG, on peut utiliser un algorithme d'aller simple car il est possible d'ordonner les noeuds. Dans le cas plus général des graphes orientés, les algorithmes d'optimisation peuvent être plus compliqués et moins efficaces pour trouver l'optimum.

Comme nous n'avons pas identifié de cas existant se rapportant exactement à nos besoins, notre objectif est de trouver d'abord la bonne manière de représenter le problème de conception d'équipement de radio logicielle multi-standards, et ensuite de trouver la meilleure manière d'associer des poids aux noeuds et aux arêtes afin d'identifier l'algorithme d'optimisation le plus intéressant pour résoudre ce problème.

3.2 Modèle de graphe théorique pour la radio logicielle multi-standards

Objectif

Notre problème de conception peut être ramené à celui d'une description à différents niveaux de granularité, donc en couches successives. Il vise à aider le concepteur à choisir le jeu d'opérateurs de traitement (chacun à un niveau de granularité idéal) nécessaires pour que l'équipement multi-standards supporte tous les cas d'utilisation prévus. L'idée sous-jacente est qu'à un certain niveau de granularité, les opérateurs peuvent être ré-utilisés plusieurs fois à l'intérieur d'une même couche protocolaire (ou standard) ou entre couches protocolaires. Le but est de fournir au concepteur des moyens d'exploration de sa concep-

tion à différents niveaux de granularité.

Définition de la structure du graphe

Afin de répondre aux besoins exprimés ci-dessus, voici le choix effectué pour la structure du graphe. Chaque noeud représente un élément de traitement (du signal) élémentaire (PE - processing element). Afin d'effectuer ce traitement, le concepteur a le choix soit de l'implanter tel quel, donc en tant qu'élément insécable (atomique), soit en invoquant des éléments plus petits d'un niveau inférieur. Dans ce cas, plusieurs éléments peuvent être nécessaires, ou plusieurs fois le même, afin d'effectuer le calcul correspondant à l'élément de niveau supérieur.

Il s'avère donc nécessaire d'utiliser un hypergraphe afin de représenter cela puisque des arêtes peuvent avoir plusieurs destinations et qu'il faut des arcs de type ET ou OU. On parle alors d'hyperarcs, comme illustré sur les Fig. 9 et Fig. 10, qui indiquent une dépendance entre noeuds de niveaux différents. Un noeud d'un niveau plus élevé (disons n) est appelé noeud parent s'il est relié à un noeud de niveau inférieur (disons $n-1$) qui est alors appelé noeud enfant ou descendant. De manière assez intuitive, un hyperarc OU signifie qu'un seul descendant est nécessaire pour effectuer le traitement équivalent au noeud parent. Dans le cas d'un hyperarc ET, tous les noeuds descendants sont nécessaires pour remplacer le noeud parent. La manière de représenter la différence est illustrée sur les 9 et 10.

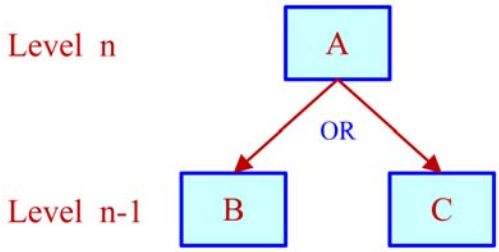

Figure 9: Hyperarc OU

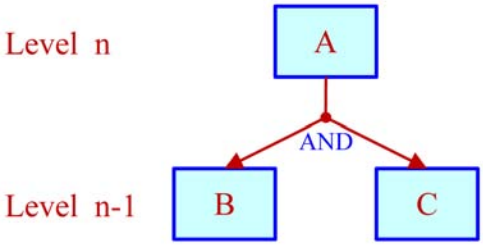

Figure 10: Hyperarc ET

La Fig. 11 représente un graphe qu'il est possible d'obtenir dans le cas d'un équipement tri-standards (simplifié). L'hypergraphe est donc constitué d'éléments de traitement de plus en plus simples au fur et à mesure que l'on descend vers le bas du graphe, les trois standards étant représentés en haut du graphe. Il est à noter qu'il n'y a pas d'assignement strict du niveau et que la seule information importante est le niveau relatif entre les noeuds qui sont reliés entre eux, pas leur niveau absolu. Cette notion n'existe pas en fait. Les arcs issus du bloc S1 montrent que la fonctionnalité de S1 peut être réalisée soit par le noeud A1, soit le noeud B1 soit le noeud A2. On voit que ces trois noeuds ne sont pas représentés au même niveau sans que cela n'ait d'incidence. En revanche, pour mettre en oeuvre le standard S3, il faut à la fois utiliser le noeud A4 et le noeud A5. Dans certains cas, un noeud peut à la fois avoir des dépendances en ET et OU, comme dans le cas du standard S2 qui peut être réalisé par le noeud A4 ou les deux noeuds A2 et A3. A titre d'exemple plus concret, pour réaliser un équipement supportant le standard S2, il est donc possible soi de réaliser le standard S2 dans un bloc unitaire donc insécable, que ce soit sous la forme d'un ASIC ou d'un programme. Lorsque tout l'équipement est conçu ainsi, c'est le cas velcro puisque si l'on veut changer de standard, il faut commuter sur un autre bloc unitaire pour la totalité des calculs associé au standard. Une autre possibilité de réalisation du standard S2 est d'implanter les noeuds A2 et A3. On remarque alors que si l'équipement passe au standard S1, A2 peut être conservé et ainsi économiser beaucoup d'efforts en termes de reconfiguration par rapport au cas velcro, par exemple en termes de temps de reconfiguration. C'est tout l'intérêt de l'approche proposée dans cette thèse. On remarque qu'il ne peut pas y a voir de cycle dans le graphe, c'est-à-dire par d'arc qui remonte, ce qui simplifie la résolution mathématique du problème.

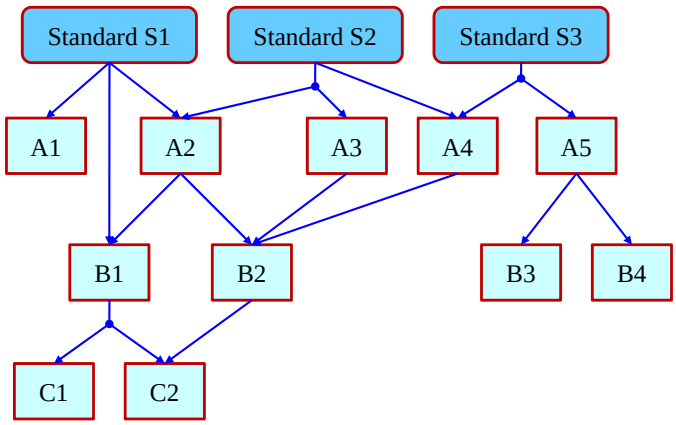

Figure 11: Graphe généralisé correspondant à un équipement de radio logicielle tri-standards

Exemple simplifié pour un équipement tri-standards WiFi, WiMAX et UMTS

La Fig. 12 illustre les mêmes principes dans le cas des standards WiFi, WiMAX et UMTS. Seul l'émetteur est représenté ici et seulement une toute petite partie de l'émetteur est véritablement prise en compte et on comprend qu'un graphe complet serait très grand

d'une part (surtout si l'on généralise aux autres couches que la couche physique), et que d'autre part, il y aurait de nombreuses manières de représenter le graphe, suivant le concepteur.

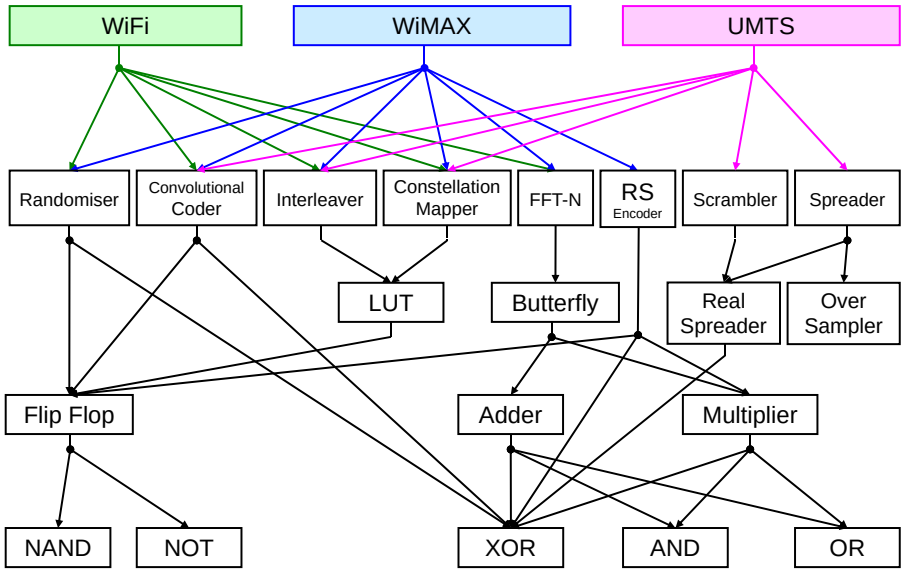

Figure 12: Structure d'un graphe pour le cas d'un équipement de radio logicielle tri-standards (émetteur simplifié)

On remarque qu'il peut y avoir autant de niveaux intermédiaires que souhaité et que "dessiner" le graphe est un travail en soi. Ce n'est pas le but de cette thèse que de développer de tels graphes complets, et les graphes qui seront utilisés le seront parce qu'ils sont fournis par d'autres études, menées par exemple afin d'identifier des opérateurs communs, ce qui est également une tache en soi, indépendante du travail de cette thèse. C'est pourquoi nous considérerons souvent des sous-graphes du graphe total, qui ne partent pas de standards en haut du graphe, et qui n'arrivent pas forcément aux opérateurs arithmétiques et bas du graphe. C'est le cas de la Fig. 13 ne représentant que les parties égalisation, channelisation et OFDM d'un équipement. De même seuls certains exemples de décomposition sont proposés. D'autres seraient possibles. Cependant, cela illustre que le nombre d'alternatives de conception peut très vite croître et que même si l'on peut estimer déduire intuitivement des choix optimaux sur de petits graphes, cela devient très difficile sur des graphes plus conséquents sans outillage.

Dans cet exemple pour la "channelisation", c'est-à-dire le filtrage et la séparation des canaux numérisés par la conversion analogique numérique en réception, une approche classique canal par canal peut être utilisée, ou une approche par bancs de filtres. Dans le cas d'un banc de filtres, on peut utiliser une FFT, ce qui est une alternative possible dans ce cas, mais de toute manière la FFT est indispensable pour la démodulation OFDM.

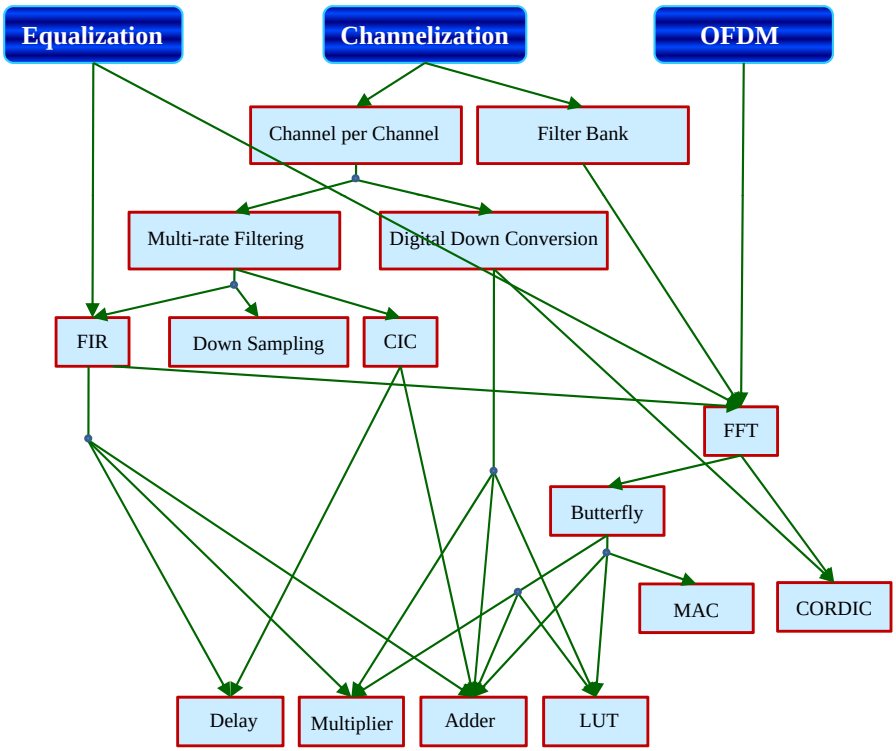

Figure 13: Exemple d'un graphe partiel

La question qui se pose alors est quelle est la meilleure décomposition ? En fonction de quel critère ? Quel est le chemin dans le graphe qui donne ce résultat ? Il est nécessaire pour cela d'ajouter des poids, autrement dit des coûts associés aux noeuds du graphe, d'en déduire une fonction de coût et d'exécuter un algorithme d'optimisation sur cette fonction de coût. C'est l'objet du prochain chapitre. Auparavant examinons brièvement une autre alternative de graphe qui a été étudiée pour résoudre ce problème.

3.3 Reformulation réseau du problème

Il s'agit de faire ici une reformulation de notre problème sous la forme d'un problème d'optimisation de réseau à source unique. Le but est de pouvoir ensuite bénéficier des nombreux algorithmes d'optimisation existant dans ce domaine.

Nous n'allons pas développer ce point dans ce résumé en français, mais nous invitons le lecteur à se référer au document en anglais. Il s'est avéré que la traduction des hyperarcs ET est compliquée et peut même devenir rédhibitoire dans certains cas. C'est pourquoi cette solution n'a pas été retenue.

4 Paramètres de coûts et coûts d'optimisation pour les systèmes de radio logicielle multi-standards

Nous explorons dans ce chapitre 4 les paramètres de coût qu'il faut associer au graphe afin de pouvoir effectuer ensuite (au chapitre 5) l'optimisation du graphe. Nous allons voir que plusieurs sortes de coûts sont envisageables. Cela dépend notamment de la manière d'implanter les traitements et de la nature de la cible matérielle les exécutant, cible dite logicielle (DSP et autres processeurs) ou dite matérielle (FPGA et ASIC). De là, une fonction de coût associée au problème de conception d'équipement de radio logicielle multi-standards est dérivée en fin de chapitre.

4.1 Paramètres de coût

Il peut y avoir de nombreux paramètres de coûts. Nous considérons dans nos travaux d'une part les deux coûts suivants associés aux noeuds : le coût de fabrication (*building cost - BC*) et le coût d'exécution (*computational cost - CC*) d'un élément de traitement (*processing element - PE*. D'autre part, sont associés aux arcs un nombre d'appels (*Number of Calls - NoC*).

Coût de fabrication

Ce paramètre concerne les noeuds du graphe. Ce coût fait référence à la dépense en termes d'implantation associée à un élément de traitement. Il sera donc à "payer" une seule fois dans l'équipement, autrement dit une fois pour toute la durée de vie de l'équipement. S'il est fait plusieurs fois appel à cet élément de traitement dans l'équipement, le coût de fabrication associé à ce traitement ne sera pas multiplié par autant. Ré-utiliser un noeud offre donc une idée de réduction du coût.

Coût d'exécution

Ce paramètre concerne les noeuds du graphe. Il y a donc deux coûts différents pour les noeuds du graphe. Ce coût est associé au temps nécessaire à l'exécution de l'élément de traitement. Il est donc multiplié par le nombre d'appels qu'il est fait à l'élément en question pendant le fonctionnement de l'équipement.

Nombre d'appels

Ce paramètre concerne les arcs du graphe. Comme évoqué précédemment, une fonctionnalité peut-être exécutée par un noeud unique ou par l'appel de un ou plusieurs noeuds de hiérarchie inférieure. Chacun de ces noeuds peut nécessiter plusieurs appels afin de pouvoir effectuer la fonctionnalité initiale. C'est cette notion que reprend le paramètre nombre d'appels.

Discussion

Il est à noter que de manière générale, un noeud de hiérarchie supérieure a un coût de fabrication plus élevé que les noeuds de hiérarchie inférieure auquel il fait appel. De même le temps d'exécution d'un noeud de hiérarchie supérieure, s'il est en général plus long que pour un seul appel à un noeud inférieur, est d'ordinaire plus faible que l'ensemble des appels et exécutions nécessaires à faire pour effectuer le même calcul à l'aide de noeuds de hiérarchie inférieure.

Prenons le cas plus concret d'une implantation matérielle ou le BC représente un coût en nombre de portes (ou surface), comme pour une implantation dans un FPGA ou un ASIC numérique. Dans l'exemple d'un élément de traitement de type filtre, il existe de nombreuses options de conception. Certaines vont chercher à paralléliser au plus leur architecture afin de gagner en vitesse d'exécution mais au prix d'un surcoût en surface. C'est le cas de l'élément de hiérarchie supérieure. En revanche, afin d'économiser en termes de surface (en coût de fabrication), il est possible de faire appel plusieurs fois à une unité de type MAC (Multiply Accumulate) qui serait alors l'élément de hiérarchie inférieure ayant un coût en surface beaucoup moins élevé (coût de fabrication), peut être aussi un coût d'exécution moins élevé aussi, mais auquel il faudra faire appel de nombreuses fois (nombre d'appels de l'arc reliant le filtre au MAC) de manière séquentielle pour effectuer un filtrage. Le coût d'exécution résultant pour le filtrage devient alors plus cher que le coût d'exécution du filtre seul. On comprend bien le compromis qu'il faut trouver entre complexité et vitesse d'exécution, entre le haut et le bas du graphe, entre granularité grossière ou fine.

La Fig. 14 reprend l'exemple de la Fig. 13 avec des valeurs de paramètres abstraits. Ils n'ont ici pas de réalité en termes de temps ou de nombre de portes, mais ils respectent une certaine logique en relatif entre paramètres de même nature. On voit q'un FIR peut être utilisé pour un coût de fabrication de 500 et un coût d'exécution de 1000. Ou alors un autre choix de conception consiste à implanter les trois opérateurs "Delay", "Multiplier" et "Adder" pour un coût total de fabrication de 15 et un coût pour chaque exécution de 8 (si on considère une exécution séquentielle à cause des dépendances de données). Mais il est nécessaire de faire appel 180 fois à cet opérateur dans cet exemple, soit un coût d'exécution total de 1440. Cette solution a donc un coût supérieur à celle consistant à utiliser le filtre directement.

4.2 Types de coût

Nous proposons une étude plus approfondie des différentes possibilités qui peuvent être pertinentes pour notre approche en termes de coûts de fabrication. Ils peuvent être considérés comme faisant partie de deux catégories :

- coût analytique
- coût d'implantation

Coût analytique

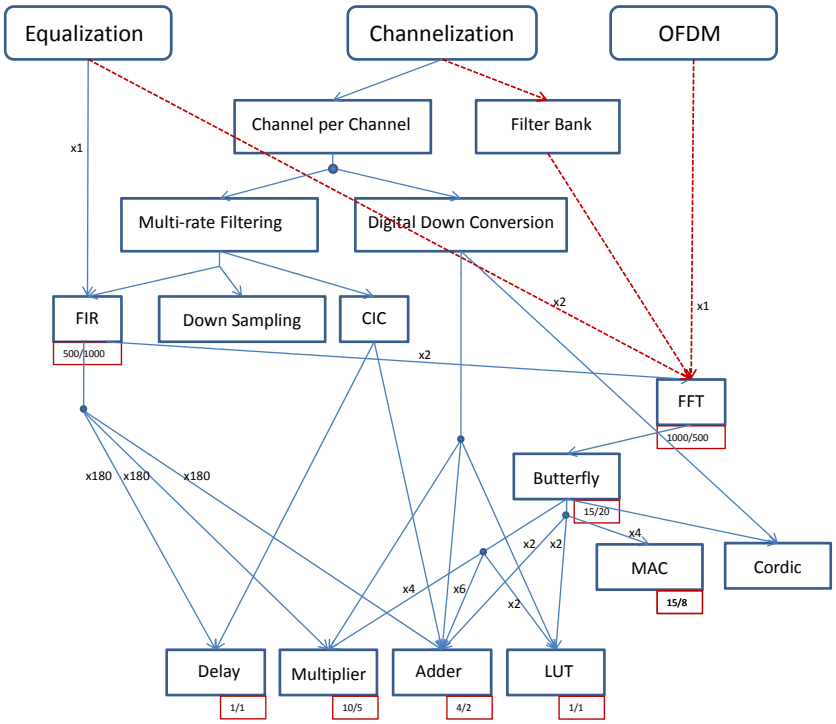

Figure 14: Exemple de graphe avec les coûts associés aux noeuds et aux arcs

Ces coûts sont issus d'une analyse de complexité des éléments de traitement composants les différents standards à implanter dans l'équipement. Cette analyse donne une estimation du nombre requis d'opérations élémentaires (par données de sortie par exemple). Les opérations élémentaires considérées peuvent être :

- Opérations arithmétiques telles que les additions et les multiplications,
- Opérations logiques,
- Opérations d'accès à la mémoire (Load/Store).

Elles ne sont pas toujours toutes prises en compte. On retrouve souvent une évaluation de complexité simplement en termes de nombre d'additions ou de multiplications. Il faut noter que le coût analytique peut être très différent du coût d'implantation, suivant l'architecture matérielle visée. Il peut même y avoir des éléments ayant un coût analytique supérieur à un autre, mais un coût d'implantation inférieur si l'architecture est propice. Les Tables en annexe B.1 et B.2 donnent des chiffres pour certains éléments de traitement du standard IEEE.802.11a et les Tables B.3 et B.4 en donnent pour le standard UMTS.

Coût de fabrication

Les plates-formes de radio logicielle requièrent la présence d'unités de traitement numériques flexibles de natures différentes en raison de la complexité des applications

qu'elles peuvent être susceptibles de supporter. Parmi ces choix, deux grandes catégories se distinguent puisqu'elles reposent sur deux domaines de calcul différents, l'un séquentiel et l'autre parallèle, et donc sur deux langages de programmation différents, le langage C et le VHDL. Certains traitements seront avantageusement exécutés par des processeurs (monde logiciel), et d'autres par des FPGA (monde matériel). Le coût de fabrication est une conséquence directe du choix de la cible d'implantation. Deux cas sont à distinguer selon qu'il s'agit d'une implantation matérielle ou logicielle.

Cas d'une implantation sur FPGA :

Le FPGA contribue à apporter au monde matériel de la flexibilité (qui n'existait pas auparavant). Cette flexibilité est jusqu'à présent surtout utilisée lors de la conception en reprogrammant (complètement) de tels composants avant de relancer une application. Les dernières familles de FPGA permettent désormais d'effectuer une reprogrammation partielle, donc d'une sous partie du FPGA (mais cela reste encore au niveau expérimental dans le domaine de la recherche [31, 32]) pendant que l'autre partie continue à fonctionner normalement. En plus de la souplesse que cela procure, un gros avantage est que le temps de reconfiguration est d'autant diminué, puisqu'il est proportionnel à la surface mise à jour. Des reconfigurations extrêmement rapides d'éléments de traitement matériels deviennent donc possibles. C'est la combinaison de la puissance de calcul du matériel (parallélisme) avec la flexibilité du logiciel (reprogrammation).

Revenons à notre propos. Quelles unités de "surface" ou de "fabrication" doit on prendre ? Dans un FPGA, on peut prendre comme coût de fabrication le nombre de slices, de CLB, de LUT, de cellules logiques, ou de bascules, etc. La Table 1 donne des chiffres pour l'implantation des standards WiFi (IEEE.802.11a) et WiMAX (IEEE.802.16-2004) [33] [34] et exprime le pourcentage d'occupation d'un FPGA donné en fonction des différentes catégories d'éléments le constituant.

Cas d'une implantation sur processeur :

Dans le monde de l'embarqué, il s'agira plutôt d'un cible de type DSP que GPP, pour des raisons de consommation. Un DSP est capable d'effectuer un certain nombre d'instructions en un seul coup d'horloge, donc en parallèle, mais ce parallélisme est limité par construction. Par exemple le C6x de Texas Instrument a 8 unités de calcul parallèle (architecture VLIW).

Le coût de fabrication peut également être appelé coût d'implantation ou coût d'achat. On peut penser à le considérer dans le cas d'un DSP, comme un coût forfaitaire.

Coût d'exécution

Le coût d'exécution est clairement fonction du temps mis pour qu'un élément de traitement fasse les calculs propres à sa fonctionnalité. Le temps d'exécution est par conséquent un bon coût d'exécution, quelle que soit la cible d'implantation. Dans le cas d'une implantation sur un processeur, le nombre de cycles nécessaires pour l'exécution d'un élément de

Table 1: IEEE 802.11a and IEEE 802.16 implementation on FPGA

Area Metrics for a XC2V3000-4FG676 Device				
	IEEE 802.11a		**IEEE 802.16-2004**	
Parameter	Used	%	Used	%
Number of Slices	1678	11	2614	18
Number of Slice Flip Flops	2353	8	3566	12
Number of 4 input LUTs	2814	9	4304	15
Number of Bonded IOBs	29	5	29	5
Number of BRAMS	12	12	12	12
Number of GCLKs	1	6	1	6

traitement peut également être considéré comme un coût d'exécution.

4.3 Formulation de la fonction de coût

Le contexte de cette étude est clairement celui d'une optimisation multi-critères, multi-objectifs. Il y a de nombreuses manières d'envisager la résolution de tels problèmes, depuis la combinaison simple des objectifs en un seul objectif, jusqu'à l'utilisation de la théorie des jeux pour coordonner l'importance relative des critères. Cependant, le flou reste quant à à la définition d'un optimum en comparaison du cas mono-critère, si bien qu'il n'est pas évident de comparer les résultats d'une méthode par rapport à une autre. On utilise des fonctions d'agrégation pour combiner plusieurs objectifs au sein d'une même fonction. Les plus courantes sont la somme pondérée, la programmation objectif, la méthode ϵ avec contrainte.

La fonction de coût à optimiser dans notre cas est composée d'une combinaison des coûts de fabrication et des coûts d'exécution. Comme déjà évoqué auparavant, ce sont deux notions a priori antagonistes puisque minimiser le coût de fabrication implique une augmentation du coût d'exécution, et vice-versa. D'où la nécessité de trouver un compromis, qui peut d'ailleurs changer suivant le concepteur ou les intérets du constructeur. En effet, suivant qu'il veut vendre des circuits ou des processeurs par exemple on comprend que les intérêts peuvent être très différents.

Développons la fonction de coût. Minimiser le coût de fabrication s'écrit :

$$\min \sum_i BC_i.N_i \tag{1}$$

avec

- BC_i représente le coût de fabrication du i^{th} élément de traitement de l'équipement,
- $N_i \in \{1, 0\}$ indique si le i^{th} noeud est présent dans l'équipement ou pas,
- $\sum_i BC_i.N_i$ est le coût total de fabrication de tous les éléments de traitement qui sont présents dans l'équipement multi-standards.

Maintenant, considérons le second objectif, c'est-à-dire minimiser le coût d'exécution. Il peut s'écrire ainsi :

$$\min \sum_k CC_k \tag{2}$$

avec

- CC_k représente le coût d'exécution du k^{th} élément de traitement de l'équipement.

Dans le contexte d'un équipement multi-standards à n standards, l'équation (2) devient :

$$\min \sum_n \sum_k CC_k((S_n)n \in N) \tag{3}$$

avec

- $((S_n)n \in N)$ indique qu'il peut y avoir n standards présents dans l'équipement avec $n = \{1, 2, \ldots, N\}$. Si l'on prend par exemple $N = 3$ alors il s'agit d'un équipement tri-standards supportant S_1, S_2 et S_3,
- $\sum_k CC_k((S_n)n \in N)$ est le coût d'exécution total pour chaque standard S_n, avec $n = \{1, 2, \ldots, N\}$.
- $\sum_n \sum_k CC_k((S_n)n \in N)$ est le coût d'exécution total pour tous les standards S_n, avec $n = \{1, 2, \ldots, N\}$.

En combinant les deux fonctions objectif des équations (1) et (3) on obtient :

$$\min \Big(\sum_i BC_i.N_i + \sum_n \sum_k CC_k((S_n)n \in N) \Big) \tag{4}$$

Mais ces deux ojectifs ne sont pas d'importance équivalente et il est par conséquent nécessaire d'introduire des poids relatifs à chaque coût. L'équation (4) devient :

$$\min \Big(\bar{\omega} \sum_i BC_i.N_i + \sum_n \sum_k \omega_n CC_k((S_n)n \in N) \Big) \tag{5}$$

avec $\bar{\omega}$ le poids octroyé au coût de fabrication de l'équipement, et ω_n est le poids d'un seul standard S_n. En effet, un seul standard est censé s'exécuter à un instant donné.
La fonction de coût à optimiser pour la conception d'un équipement de radio logicielle est donc :

$$\mathbf{C_{SDR}} = \min_{bool((S_n)n \in N)} \Big(\bar{\omega} \sum_i BC_i.N_i + \sum_n \sum_k \omega_n CC_k((S_n)n \in N) \Big) \tag{6}$$

La contrainte $bool((S_n)n \in N)$ vérifie si tous les standards sont implantés dans l'équipement.

Le coût à n'importe quel niveau du graphe peut être calculé ainsi :

$$Cost_{level} = \Big(\sum_i CC_i.NoC_i \Big).NoC_{i+1} + \sum_i BC_i.N_i \tag{7}$$

Nous présenterons dans le chapitre suivant comment effectuer l'optimisation de ce problème.

4.4 Interface graphique

Une interface graphique a été développée en langage Java afin de faciliter la construction des graphes et des paramètres de coûts associés aux arcs et aux noeuds. La Fig. 15 en montre une capture d'écran.

Les fonctionnalités sont les suivantes :

- ***Create Processing Element***: pour insérer un nouveau noeud dans le graphe.
- ***Delete Processing Element***: pour supprimer un noeud.
- ***Create AND hyperarc***: pour créer entre des noeuds un arc ET. Il faut pour cela sélectionner dans des listes un noeud parent et des noeuds enfants.
- ***Create OR hyperarc***: idem que précédemment, pour un arc OU.
- ***Delete hyperarc***: pour supprimer un arc.
- ***Add Parameters***: ouvre la fenêtre de gestion des paramètres des noeuds et des arcs.

Le graphe peut être sauvegardé dans un format qui après être "parsé" peut être utilisé comme entrée du code d'optimisation.

5 Technique d'optimisation

Nous avons formulé dans le précédent chapitre la fonction de coût bi-objectifs. Dans ce chapitre 5, nous allons étudier les différentes techniques pour résoudre le problème d'optimisation associé à la conception d'équipements de radio logicielle multi-standards tel que nous l'avons formulé. La résolution du problème par une solution exacte requerrait une complexité exponentielle avec le nombre de noeuds considérés, ce qui n'est possible en

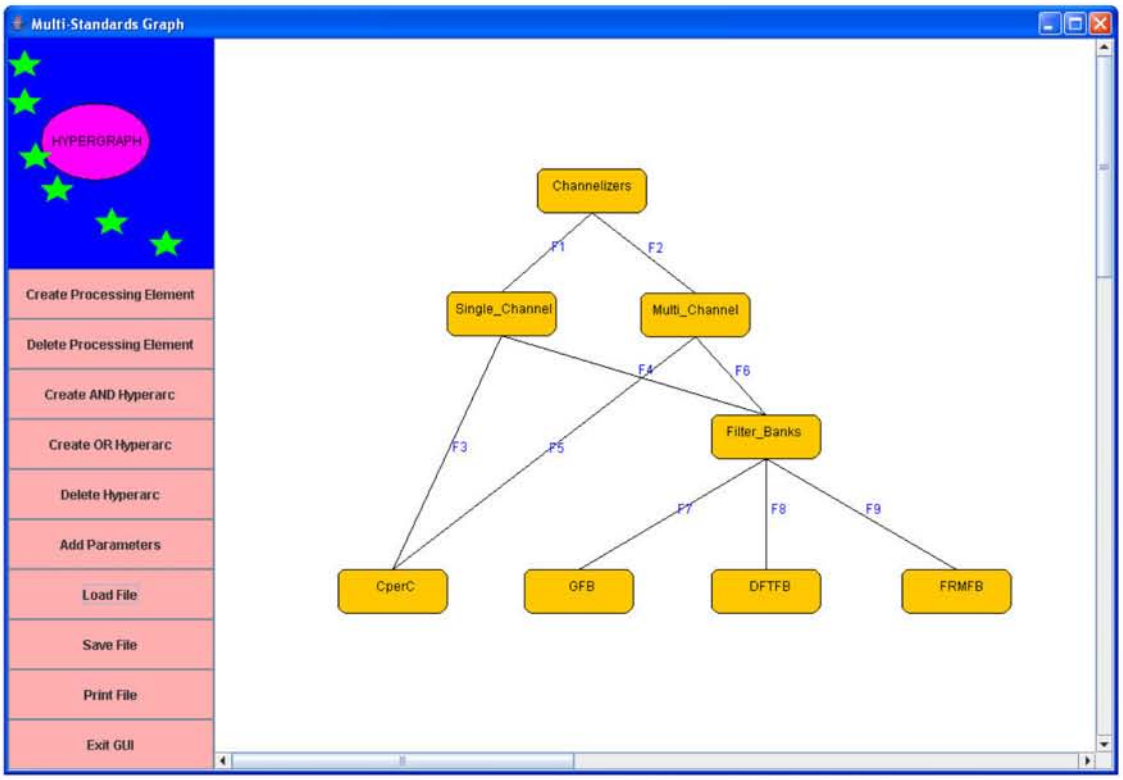

Figure 15: Capture d'écran de l'interface graphique développée pour contruire les graphes

un temps acceptable que pour un nombre de noeuds restreint même sur les calculateurs les plus rapides. C'est pourquoi des méthodes sous-optimales sont considérées. Nous allons parcourir différentes solutions possibles et discuter de leurs avantages et inconvénients.

5.1 Les techniques d'optimisation

Les techniques d'optimisation peuvent être classées en deux grandes familles :

- déterministe,
- stochastique.

La Fig. 16 illustre diverses techniques d'optimisation en les classifiant suivant ces deux catégories.

Nous ne parcourrons pas dans ce résumé l'ensemble des cas évoqués dans la Fig. 16.

5.2 La recherche exhaustive

Une recherche exhaustive consiste à parcourir toutes les solutions de l'espace d'exploration architecturale afin d'en trouver la meilleure. Cette approche n'est malheureusement pas

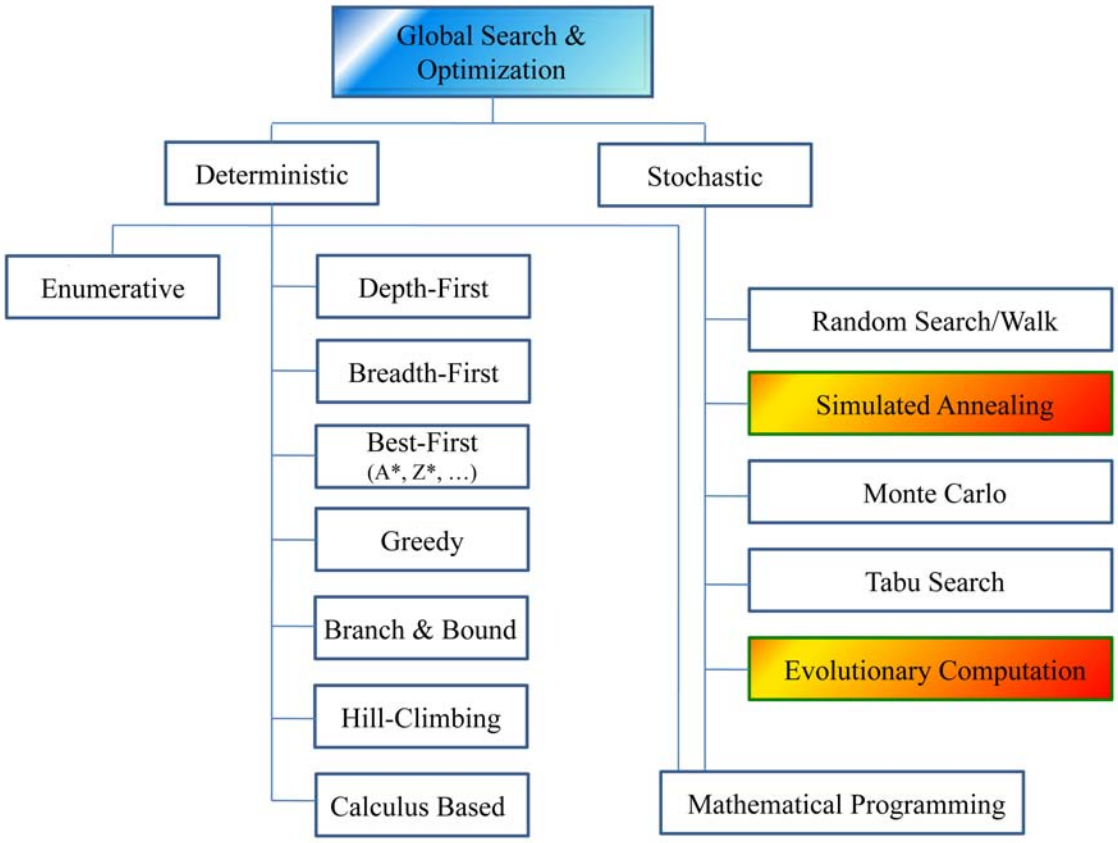

Figure 16: Approches d'optimisation [35]

faisable lorsque l'espace d'exploration devient grand, ce qui est le cas dans la réalité d'une conception d'équipement de radio logicielle multi-standards.

Réfléchir à la mise en oeuvre d'une telle approche présente l'intérêt de devoir trouver le moyen de parcourir de manière systématique l'ensemble des cas possibles de conception. La Fig. 17 montre comment une telle recherche est effectuée. D'abord les noeuds et les coûts sont initialisés. Pour les noeuds, cela revient à les classer par ordre, ainsi que les catégoriser en tant que noeuds utiles, inutiles, hauts ou bas. Les noeuds hauts sont des branches et les noeuds bas sont des feuilles.

Une fois que les noeuds sont classés, le coût total de l'équipement est calculé à partir des noeuds utiles. Puis un noeud est retiré de l'ensemble des noeuds et la faisabilité du graphe est vérifiée et le coût est mis à jour. Si le nouveau coût est inférieur au coût initial, il est accepté. Et ainsi de suite, les autres noeuds sont éliminés jusqu'à ce que le coût minimal donnant un graphe valide soit obtenu. Cela donne la solution optimale ayant le coût minimal.

Considérons le cas du graphe de la Fig. 18 où l'on a représenté des coûts que sur certains noeuds et arcs afin de garder le problème suffisamment simple pour être résolu

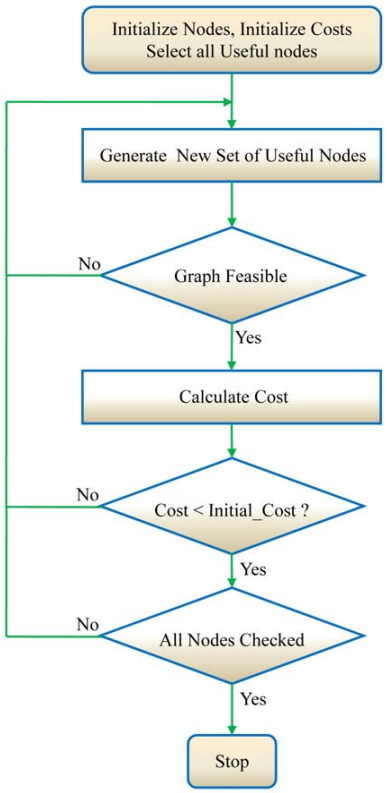

Figure 17: Algorithme de la méthode par recherche exhaustive

par la méthode de la recherche exhaustive.

A titre d'exemple, considérons le noeud FFT car il est simple de voir qu'il permet l'implantation de l'égaliseur, du "channelizer" et de l'OFDM. Il existe peut-être d'autres solutions moins évidentes, mais c'est le rôle d'un programme de parcourir toutes les possibilités, au-delà de l'intuition. La fonction de coût pour le noeud FFT peut être extraite de l'équation (7) rappelée ici :

$$Cost_{level} = \Big(\sum_{i=1}^{i=m} CC_i.NoC_i \Big) \times NoC_{i+1} + \sum_{i=1}^{i=m} BC_i.N_i \tag{8}$$

$$Cost_{FFT} = CC_{FFT} + BC_{FFT} \tag{9}$$

soit

$$Cost_{FFT} = 50 + 3000 = 3050 \tag{10}$$

Ou en utilisant le niveau inférieur, soit l'opérateur papillon ou Butterfly (on ne considère pas le cas de l'opérateur Cordic dans ce petit exemple) :

$$Cost_{FFT-1} = CC_{Butterfly}.NoC_{Butterfly} + BC_{Butterfly} \tag{11}$$

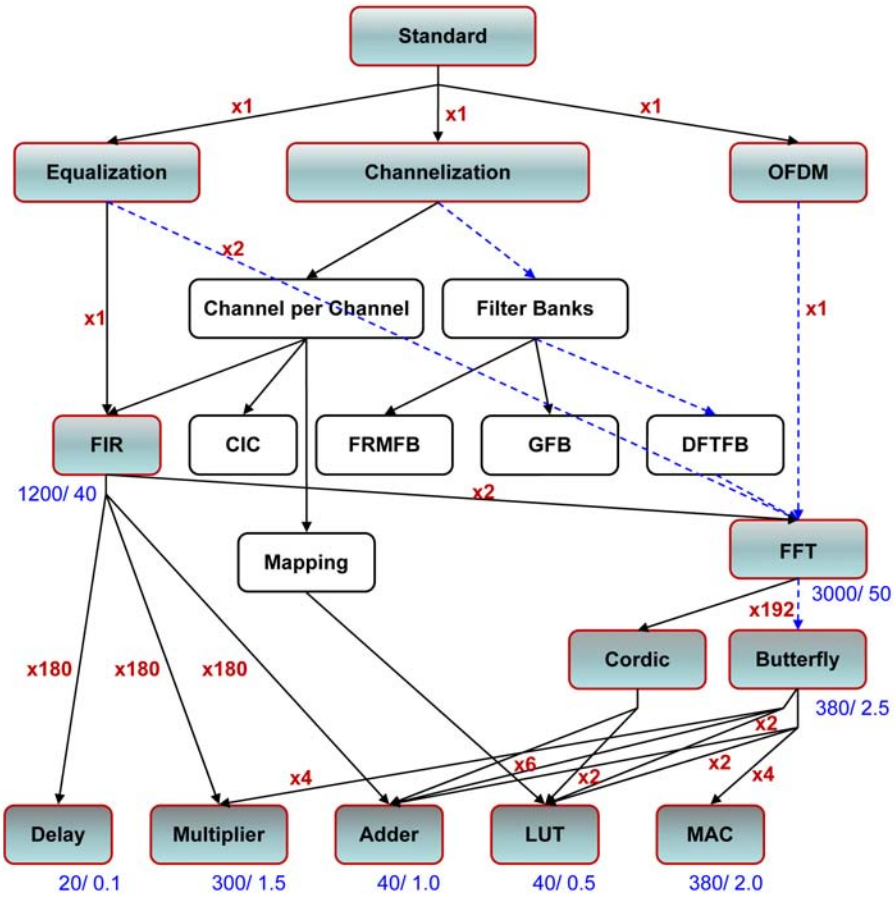

Figure 18: Exemple de graphe avec les coûts associés

soit

$$Cost_{FFT-1} = 2.5 \times 192 + 380 = 860 \tag{12}$$

Ou en utilisant le niveau inférieur au Butterfly, soit les opérateurs MAC, LUT et Adder ou les opérateurs LUT, Adder et Multiplier :

$$Cost_{\{(FFT-2)\}} = \Big(\sum_{i=1}^{i=3} CC_i NoC_i\Big) \times NoC_{i+1} + \sum_{i=1}^{i=3} BC_i.N_i \tag{13}$$

soit

$$\begin{aligned} Cost_{\{(FFT-2)a\}} &= \big(4 \times CC_{MAC} + 2 \times CC_{LUT} + 2 \times CC_{Adder}\big) \times 192 \\ &+ BC_{MAC} + BC_{LUT} + BC_{Adder} \end{aligned} \tag{14}$$

ou encore

$$Cost_{\{(FFT-2)a\}} = \big(4 \times 2 + 2 \times 0.5 + 2 \times 1\big) \times 192 + 380 + 40 + 40 = 2572 \tag{15}$$

En utilisant la deuxième option :

$$\begin{aligned} Cost_{\{(FFT-2)b\}} &= \big(4 \times CC_{Multiplier} + 6 \times CC_{Adder} + 2 \times CC_{LUT}\big) \times 192 \\ &+ BC_{Multiplier} + BC_{Adder} + BC_{LUT} \end{aligned} \tag{16}$$

soit

$$Cost_{\{(FFT-2)b\}} = \big(4 \times 1.5 + 6 \times 1 + 2 \times 0.5\big) \times 192 + 300 + 40 + 40 = 2876 \tag{17}$$

Ainsi, si l'on considère que l'équipement est réduit à une FFT, on obtient la fonction de coût suivante :

$$\begin{aligned} \mathbf{C_{FFT}} &= \min\big(Cost_{FFT}, Cost_{\{FFT-1\}}, Cost_{(\{FFT-2)a\}} \\ &, \quad Cost_{\{(FFT-2)b\}}\big) = \mathbf{Cost_{\{FFT-1\}}} \end{aligned} \tag{18}$$

La méthode de recherche exhaustive conclut que c'est le papillon qui est le meilleur opérateur commun au sens du coût le moins élevé pour le graphe de la Fig. 18. Notons que dans la fonction de coût, les poids $\bar{\omega}$ et ω_n sont neutralisés à 0,5 chacun et ce sera toujours le cas par la suite, sauf indication contraire.

5.3 Le recuit simulé

La méthode d'optimisation appelée recuit simulé repose sur le principe physique du refroidissement lent vers un minimum d'énergie. C'est une méthode d'obtention de solution sous-optimale adaptée dans des cas d'optimisation où l'espace d'exploration est très large. Cela consiste en un parcours aléatoire de l'espace des solutions et si un déplacement améliore l'optimum, donc réduit le coût de la fonction de coût, il est poursuivi. Si un déplacement augmente le coût, le déplacement peut-être poursuivi dans le même sens avec une probabilité $p < 1$, dans le but d'éviter les minimums locaux. Cette probabilité décroît exponentiellement ou avec le temps ou avec la quantité selon laquelle le coût a augmenté. De même que si la température décroît suffisamment lentement alors le métal se rigidifie dans une structure cristalline à minimum d'énergie, si la probabilité de mouvement varie lentement, c'est que l'optimum est obtenu.

Supposons que la fonction V définie sur un jeu fini de possibilités S soit à minimiser. Supposons que pour chaque état s de S, il y a un jeu $N(s)$, avec $N(s) \subset S$, que nous appelons voisins de s. Soit la matrice de probabilités de transition R sur S telle que $R(s, s^{'}) > 0$ si et seulement si $s^{'}$ est compris dans $N(s)$.

Soit $T_1, T_2, \ldots$ une séquence de nombres strictement positifs tels que :

$$T_1 \geq T_2 \geq \ldots \quad et \quad \lim_{k \to \infty} T_k = 0 \tag{19}$$

Nous considérons l'algorithme suivant pour construire une séquence d'états $X_0, X_1, \ldots$ Un état initial X_0 est choisi. Etant donné qu'un état suivant potentiel est choisi parmi $N(s)$ avec une distribution de probabilité telle que :

$$P[Y_k = s^{'} | X_k = s] = R(s, s^{'}) \tag{20}$$

Alors nous établissons

$$X_{k+1} = \begin{cases} Y_k & avec \quad P_k \\ X_k & sinon \end{cases} \tag{21}$$

avec

$$P_k = \left[\frac{-[V(Y_k) - V(s)]}{T_k} \right] \tag{22}$$

$$\lim_{k \to \infty} P[X_k \in S^*] = 1 \tag{23}$$

L'équation (21) spécifie comment la séquence $X_1, X_2, \ldots.$ est choisie. L'idée du recuit simulé est d'essayer d'obtenir (23) en faisant tendre T_k vers zéro quand k tend vers l'infini, avec S^* qui représente le jeu d'états de S pour lesquels V atteint sa valeur minimale.

La mise en oeuvre de l'algorithme de recuit simulé est illustrée par la Fig. 19. les éléments suivants sont nécessaires :

- une représentation des solutions possibles,
- un générateur de changements aléatoires dans les solutions,
- un moyen d'évaluation des fonctions problèmes
- un plan de refroidissement - une température initiale et des règles pour la faire décroître pendant la recherche

Un programme de recuit simulé a été développé en langage C++. L'exécution du programme sur l'exemple de la Fig. 18 retrouve bien la solution révélée par la recherche exhaustive, à savoir le papillon comme opérateur commun à ce sous-graphe. C'est une preuve du bon choix de l'algorithme de recuit simulé et de sa bonne programmation.

5.4 Les algorithmes génétiques

Les algorithmes génétiques font partie de la famille des modèles de calcul inspirés par l'évolution. Ils visent à traduire un problème sous la forme d'une structure de chromosomes et comme indiqué sur la Fig. 20. Ils appliquent les lois naturelles de l'évolution pour combiner, sous forme itérative, les solutions d'une population de chromosomes, à savoir : la sélection, les croisements et les mutations. Ces algorithmes sont utilisés pour résoudre de nombreux problèmes, de natures extrêmement variées.

Initialisation

Une population initiale est tirée au hasard. Elle est constituée de chromosomes constitués chacun d'un jeu de gènes. On peut calculer un coût pour chaque chromosome et les classer par ordre croissant de la fonction de coût qui leur est associée. La Fig. 21 donne une représentation simplifiée de quelques chromosomes d'une population dont les gènes

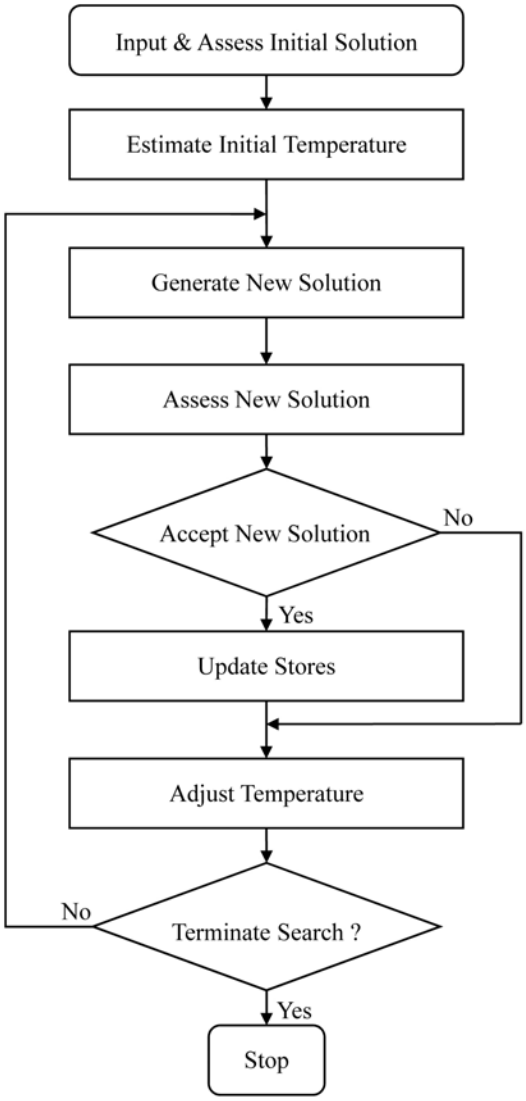

Figure 19: Algorithme du recuit simulé

prennent leur valeur dans {0, 1}.

Sélection

A chaque génération, une sélection est effectuée avec une probabilité en relation avec le coût de chacun des membres de la population.

Croisements

Comme indiqué sur la Fig. 22, des gènes de deux chromosomes de la génération précédente sont combinés, comme les parents biologiques mêlent leur gènes lorsqu'ils ont un

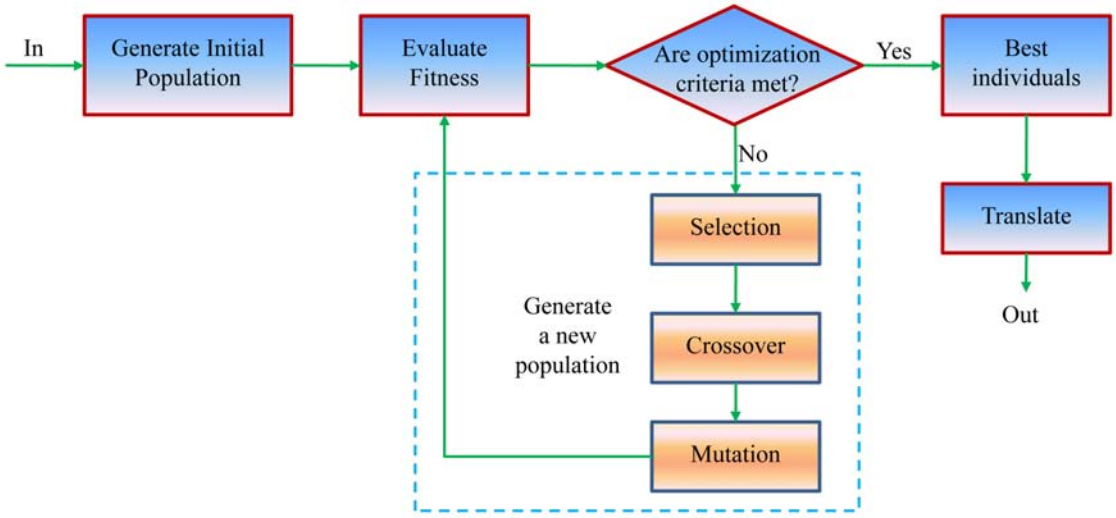

Figure 20: Schéma simplifié d'un algorithme génétique

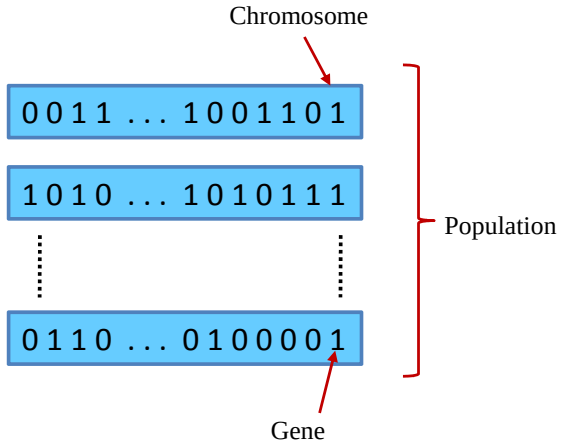

Figure 21: Représentation simplifiée d'une population

enfant.

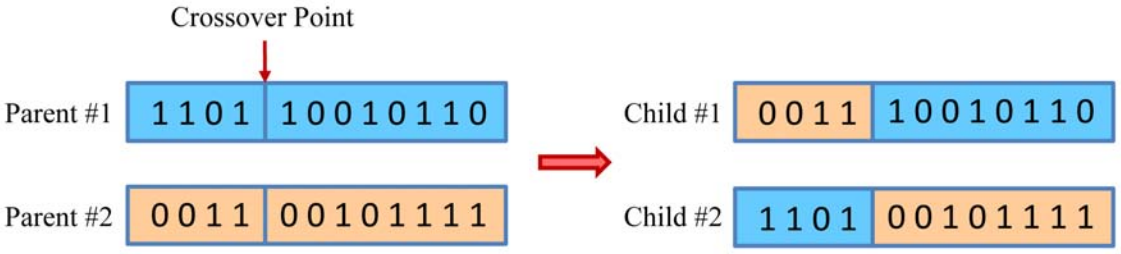

Figure 22: Croisements

Mutations

Des gènes sont arbitrairement modifiés dans des chromosomes afin d'amener de nouveaux motifs dans la population, comme illustré sur la Fig. 23. Cela permet d'éviter des minimums locaux associés au tirage de la population initiale.

Figure 23: Mutations

Déroulement

Les générations se succèdent et cela conduit peu à peu à la production de chromosomes minimisant la fonction de coût associée au problème. La problématique des algorithmes génétiques consiste à trouver le nombre de générations qu'il faut faire succéder, la proportion entre mutations, croisements et sélection, etc.

5.5 Application

Afin de mieux évaluer les trois techniques d'optimisation exposées précédemment, confrontons les à l'exemple générique de la Fig. 24. Les valeurs des paramètres BC, CC et NoC ont été prises arbitrairement et seul un jeu limité de 8 noeuds en possèdent afin de simplifier le cas d'étude.

Les résultats de la recherche exhaustive sont montrés sur la Fig 25 où sont représentés les coûts des solutions obtenues en fonction du nombre d'itérations de l'optimisation. Une solution est bonne si elle est meilleure que celles trouvées auparavant. Les colonnes sombres sont de bonnes solutions et la ligne en pointillé montre la tendance. Mais dans la recherche exhaustive, tous les cas possibles sont parcourus donc la progression n'est pas régulière et la meilleure solution pourrait être trouvée (para hasard) très rapidement, mais cela n'arrêterait pas l'algorithme qui doit aller à son terme.

La Fig. 26 reprend le même principe que la figure précédente dans le cas du recuit simulé. Comme on s'y attendait l'algorithme converge plus rapidement que le cas précédent de la recherche exhaustive. En fait la recherche exhaustive sert de référence pour voir si les autres algorithmes envisagés convergent bien vers la bonne solution. Ces résultats ont été obtenus en exécutant le programme développé.

L'étude d'un algorithme génétique est complexe et aurait nécessité un travail qui va au-delà de cette thèse. Cependant, nous avons évalué cette possibilité avant de faire un choix d'algorithme. Dans l'exemple de la Fig. 27, seulement 10 chromosomes ont été

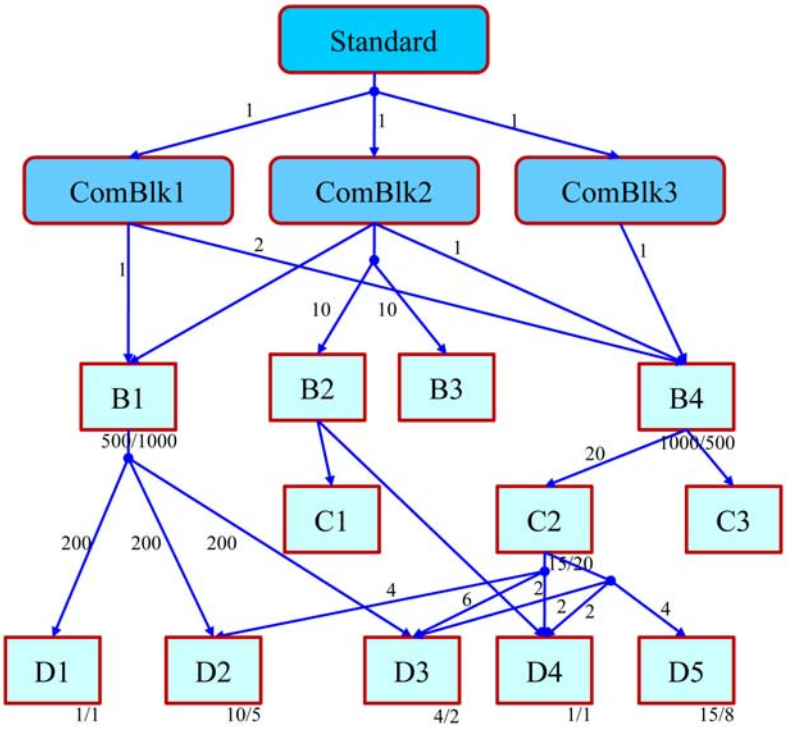

Figure 24: Equipment SDR générique

choisis et seulement sur 10 générations.

La Fig. 28 représente les élements retenus à chaque itération. elle montre que si l'équipement est fabriqué à l'aide de l'opérateur C2, c'est la meilleure solution, pour un coût de 1615 (en haut de la figure).

Discussion

Le recuit simulé peut être vu comme un algorithme génétique avec une population de 1 [36]. Il n'y a alors que des mutations, pas de croisement. Toutes deux reposent sur le principe que les bonnes solutions ont plus de chances d'être trouvées près des bonnes solutions déjà connues plutôt que de tirer des cas au hasard dans l'ensemble de l'espace des solutions. Si les mutations sont dominantes alors l'algorithme génétique agit de manière "parallèle" au recuit simulé, avec des solutions améliorées indépendamment. Pour les problèmes où les meilleures solutions sont proches cela donne un avantage au recuit simulé quand les solutions basées sur des recombinaisons ne sont pas efficaces.

Le temps de convergence d'un algorithme dépend du nombre d'itération nécessaire pour obtenir la solution au problème. Bien sûr les heuristiques telles que le recuit simulé et les algorithmes génétiques convergent plus rapidement que la recherche exhaustive, c'est leur raison d'exister. Dans le cas d'un algorithme génétique, le point de convergence sera

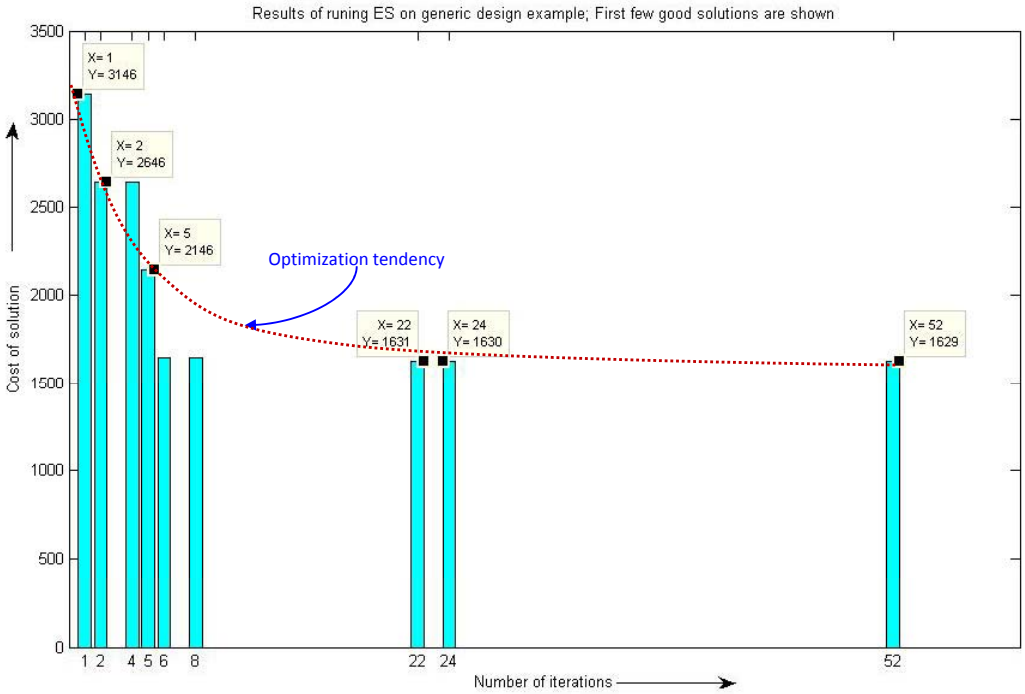

Figure 25: Resultats de la recherche exhaustive sur l'exemple générique

obtenu après un nombre plus important d'itérations que pour le recuit simulé.

Notons par ailleurs au-delà de cette conclusion de nombreuses comparaison de ces deux solutions ont été effectuées dans la littérature donnant l'avantage au recuit simulé. Cependant il est bien sûr impossible de dire arbitrairement que le recuit simulé est meilleur que les algorithmes génétiques en général. Cela dépend du problème à optimiser. Dans [37] des problèmes de minimisation de coûts d'arbres ont été confrontés notamment à des algorithmes génétiques et de recuit simulé. Les résultats montrent que pour un même temps d'exécution, le recuit simulé donne de meilleurs résultats. Pour des problèmes de transport-allocation il est montré dans [38] que dans tous les cas, les résultats basés sur le recuit simulé sont meilleurs que toutes les autres techniques. D'autre part, une comparaison directe a été effectuée entre du recuit simulé adaptatif et des algorithmes génétiques disponibles publiquement, sur des problèmes pour lesquels les algorithmes génétiques sont adaptés et adoptés. Dans tous les cas, le recuit simulé a battu l'algorithme génétique [39]. D'une manière générale les algorithmes génétiques sont très consommateurs en mémoire car il faut stocker les états intermédiaires de toutes les populations [40].

5.6 Conclusion sur le choix de l'algorithme d'optimisation

A la suite de l'étude de la bibliographie, ainsi que de la structure du graphe et des différentes possibilités en termes d'algorithmes d'optimisation, le choix a été fait d'utiliser un algorithme de recuit simulé qui a été programmé en C++. Le programme prend en

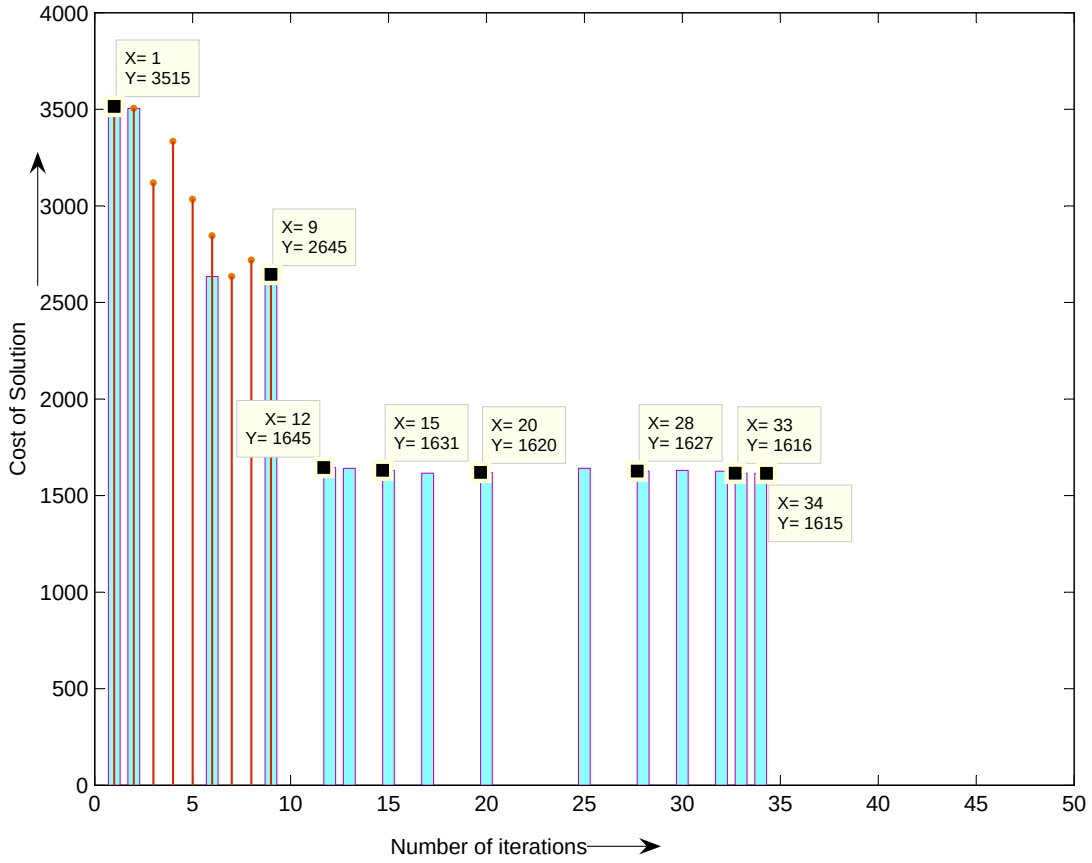

Figure 26: Resultats du recuit simulé sur l'exemple générique

entrée une version texte du diagramme dessiné dans l'interface graphique Java qui a été développée. Comme nous le verrons par la suite dans les chapitres 6, 7, 8 et surtout 9, le choix d'un algorithme de recuit simulé a complètement répondu à nos besoins à l'échelle de cette thèse. En outre, aucun signe de remise en cause ne nous est apparu à l'échelle des graphes que nous avons optimisé, ce qui nous rend confiants pour son utilisation sur des graphes beaucoup plus conséquents. Cependant, nous n'excluons pas que des travaux futurs puisse trouver une meilleure solution, surtout si d'autres contraintes sont ajoutées au problème.

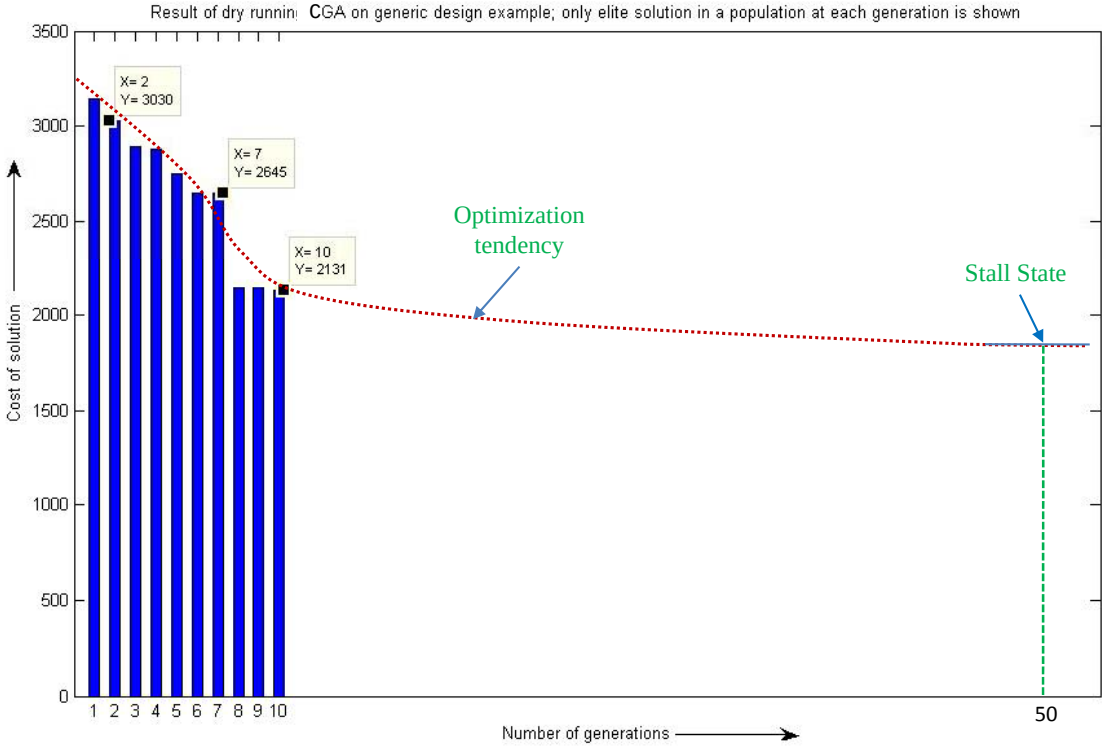

Figure 27: Resultats d'un algorithme génétique sur l'exemple générique

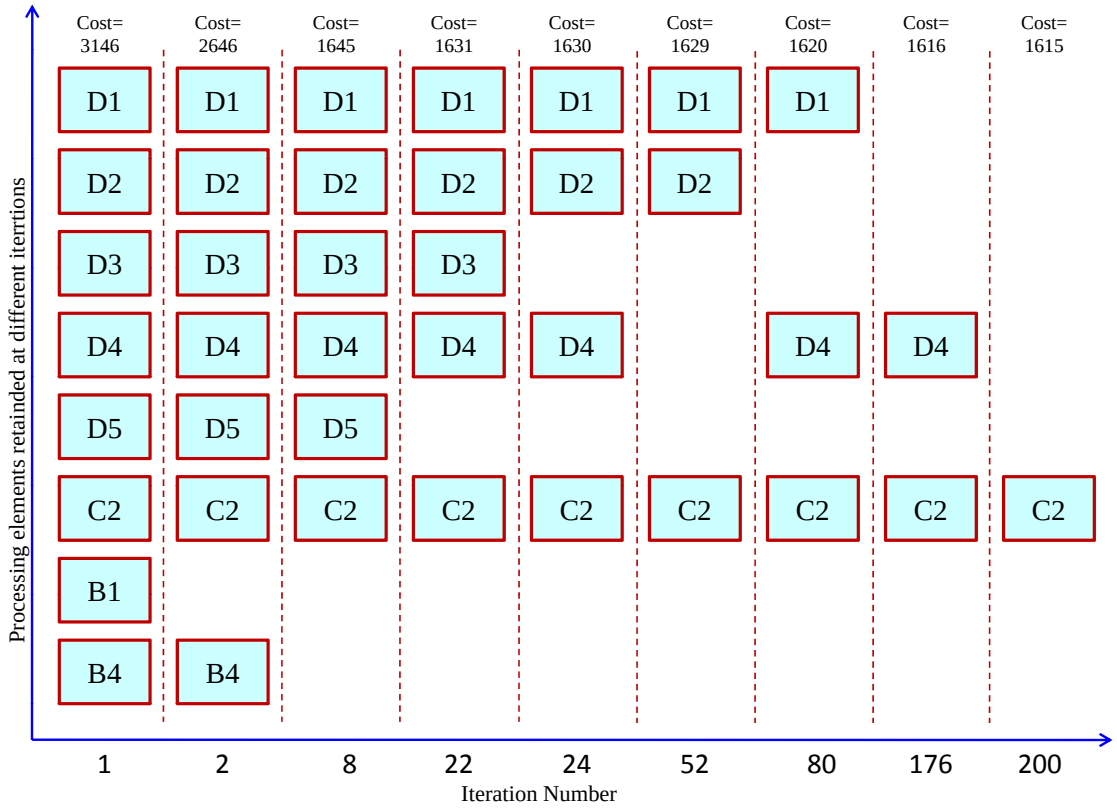

Figure 28: Elements retenus aux différentes itérations

Introduction à la partie III

Lors des chapitres précédents, il a été discuté de l'approche de conception par opérateurs communs pour les équipements multi-standards, puis de la manière de la représenter sous la forme d'un graphe. Plusieurs techniques d'optimisation ont été alors passées en revue afin d'obtenir une solution de conception optimale. Cette partie illustre par des exemples la mise en oeuvre de la méthode de conception proposée dans ce manuscrit en la confrontant notamment à des travaux effectués dans d'autres recherches menées afin d'identifier des opérateurs communs. Les quatre chapitres de cette partie portent donc sur trois candidats potentiels d'opérateurs communs et la mise en commun de deux d'entre eux dans un exemple à plus large échelle. Les chapitres 6, 7 et 8 reposent sur une collaboration avec trois autres doctorants de l'équipe, chacun d'entre eux ayant travaillé sur l'opérateur commun considéré dans le chapitre. Notre travail met notamment en perspectives leurs résultats en termes d'implantation sur cible FPGA grâce à l'approche d'optimisation graphique proposée ici.

Le chapitre 6 présente une étude portant sur l'utilisation de la FFT comme opérateur commun. Dans le chapitre 7, c'est l'opérateur LFSR qui est étudié. Puis le chapitre 8 porte plus particulièrement sur le front-end numérique et le filtrage de canalisation autour de l'opérateur FRMFB. Enfin, ce problème est étendu dans le chapitre 9 à un exemple de conception plus large afin de mieux discuter de l'approche de conception proposée dans sa globalité.

6 Cas d'étude : opérateur commun FFT dans un équipement de radio logicielle multi-standards

La transformée de Fourier est réputée pour son utilisation dans la conversion d'un signal temporel en un signal fréquentiel [41]. Mais elle a bien d'autres vertus et a particulièrement gagné en intérêt grâce à l'introduction de la FFT (Fast Fourier Transform) par Cooley et Tukey [42]. Cela a permis de décroître la complexité d'exécution , permettant ainsi de l'utiliser de manière efficace dans le domaine numérique. La complexité d'une FFT brutale est de N^2 multiplications complexes et $N.(N-1)$ additions complexes, et elle passe à $N.log_2N$ multiplications et additions complexes si N est la longueur de la transformée.

Partant du constat qu'on retrouve la FFT dans de nombreux cas dans les chaînes de communications numériques, [17] a considéré la FFT comme opérateur commun potentiel. Ajoutons qu'au-delà des traitements spécifiques au traitement de la radio, la FFT

peut aussi être utilisée dans des algorithmes de traitement de la couche applicative (par exemple vidéo), ce qui entre en compte dans l'approche de conception inter-couche qui est envisagée ici.

Dans ce contexte, il peut être intéressant d'envisager d'effectuer dans le domaine fréquentiel des calculs qui le sont habituellement dans le domaine temporel afin de bénéficier de la présence de la FFT. Dans ce chapitre 6, nous considérons le cas de l'utilisation de la FFT pour le codage de canal et plus particulièrement pour le codage de Reed Solomon, issu des travaux de [43]. Ces travaux ont permis de mettre en évidence qu'il est possible d'utiliser une FFT paramétrable capable d'effectuer les traitements habituels d'une FFT dans $\mathbb{C}$, mais aussi ceux du codage RS dans GF. Cela est possible en fait pour un sous-ensemble des codes de RS, ceux définis dans le corps $GF(F_t)$ alors que les codes de RS habituels sont définis dans $GF(2^m)$. La Fig. 29 illustre comment un tel opérateur peut être utilisé dans ces différents cas. Elle détaille aussi que l'algorithm d'encodage/décodage RS inclut les étapes de calcul du syndrome, de recherche de Chien, d'un calcul polynomial, de l'algorithme de Berlekamp Massey et celui de Forney. La Fig. 29 montre que parmi ceux-ci, les algorithmes de recherche de Chien et de calcul de syndrome, qui sont les plus complexes, peuvent être exécuté par l'opérateur FFT reconfigurable proposé.

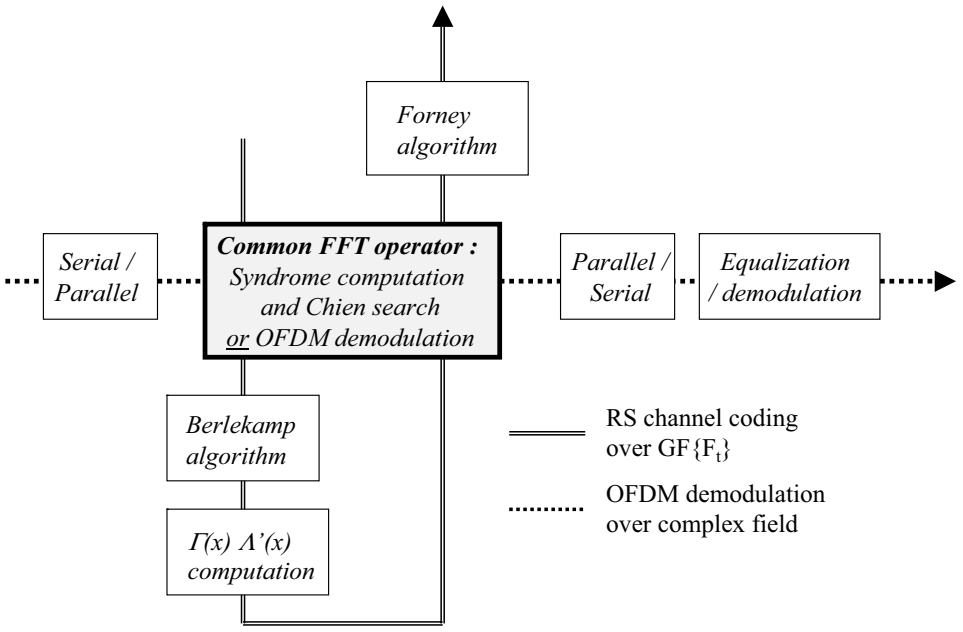

Figure 29: Exemple d'une FFT partagée entre un démodulateur OFDM et un décodeur RS dans $GF(F_t)$.

6.1 Opérateur DMFFT

Cet opérateur commun de FFT reconfigurable a été nommé DMFFT (Dual Mode FFT) [19]. Il permet à la fois d'opérer des transformations de Fourier dans $\mathbb{C}$ appelées FFT, et des transformées de Fourier dans le corps de Galois de taille Ft $GF(F_t)$ appelées FNT

(Ft est pour Fermat). Son implantation repose sur le papillon reconfigurable (RBPE - Reconfigurable Butterfly Processing Element) de la Fig. 30.

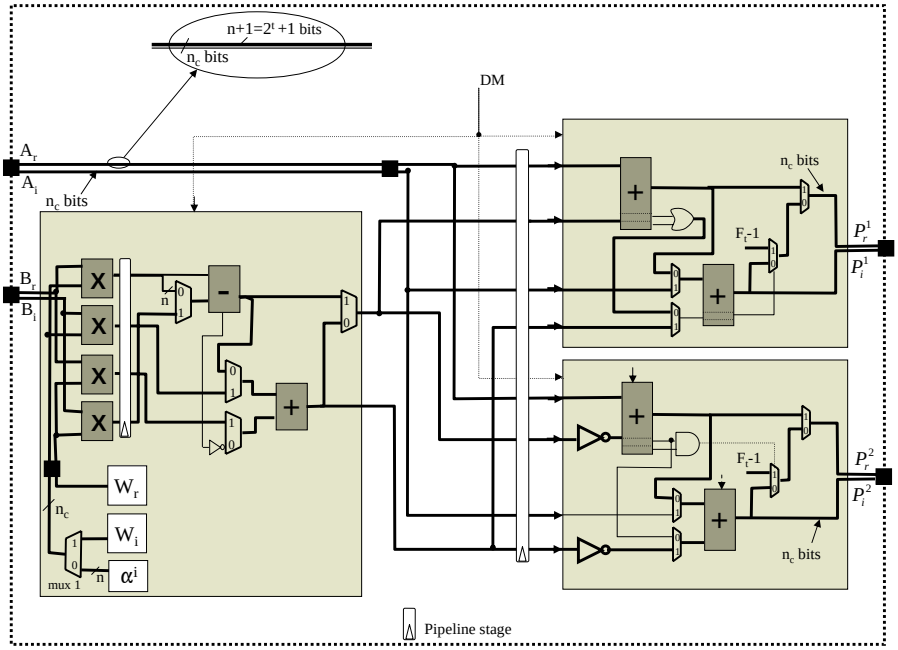

Figure 30: Architecture du papillon reconfigurable

Si l'on compare un équipement utilisant l'opérateur reconfigurable DMFFT ou une approche de type velcro pour effectuer la FFT et le codage RS, voici ce que l'on obtient. La Table 2 relève les résultats après implantation sur un composant FPGA de type Stratix-II et pour des mots de différentes longueurs n_c, d'une part de la DMFFT-64, d'autre part de la FFT-64 et de la FNT-64 (velcro operator). Les résultats sont en faveur de la DMFFT-64, aussi bien en termes de nombre d'ALUT (adaptive look-up table), de vitesse d'exécution, d'occupation mémoire, que de taux performance sur coût. Le nombre de ALUT est relatif à l'occupation en nombre de portes. Le taux performance sur coût $\eta = \frac{1}{T.C} * 10^6$ issu de [44], avec T est le temps d'exécution en nanosecobdes (ns) et C le nombre de blocks logiques relativement à un circuit FPGA. Quand η augmente, un meilleur compromis entre temps d'exécution et coût est obtenu.

La table 3 compare la DMFFT-256 avec la solution velcro FFT/FNT-256. On retrouve les mêmes avantages en termes de mémoire et des résultats toujours positifs mais moindre pour les autres paramètres. Cela illustre l'intérêt d'avoir recours à des opérateurs communs plutôt que des solutions velcro dans la conception, en terme de complexité de réalisation. Un autre avantage primordial en radio logicielle est que dans le cas d'une approche velcro, le temps de reconfiguration est plus pénalisant que dans celui d'un changement de paramètres de la DMFFT. En dehors d'un contexte de radio intelligente avec un hand-over vertical, cela n'a pas d'importance dans le cas d'une utilisation inter-standards d'un même opérateur puisque nous faisons l'hypothèse d'une utilisation séquentielle des standards avec un temps non contraint entre deux communications. Mais dans le cas où l'opérateur

Table 2: Comparaison entre la DMFFT-64 et la FFT/FNT-64 Velcro sur un Stratix-II, EP2S15F484C3

n_c	9	10	12	14	16
Velcro operator	4205 ALUTs 4.86 ns	4768 ALUTs 5.27 ns	5831 ALUTs 5.46 ns	6844 ALUTs 5.81 ns	8143 ALUTs 6.62 ns
DMFFT	3109 ALUTs 4.78 ns	3744 ALUTs 4.97 ns	4857 ALUTs 5.45 ns	5975 ALUTs 5.85 ns	7387 ALUTs 6.65 ns
Memory saving	33 %	31 %	27.2 %	24.3 %	21.9 %
ALUT's gain	26 %	21.4 %	16.7 %	12.7 %	9.2 %
η's gain	37.4 %	35 %	20 %	13.9 %	9.7 %

est partagé entre plusieurs fonctionnalités d'un même standard, cela est crucial puisque l'on va effectuer à un rythme très élevé des changement de fonctionnalité de l'opérateur.

Table 3: Comparaison entre la DMFFT-256 et la FFT/FNT-256 Velcro sur un Stratix-II, EP2S15F484C3

n_c	9	10	12	14	16
Velcro operator	5327 ALUTs 5.02 ns	6079 ALUTs 4.95 ns	7518 ALUTs 5.45 ns	8966 ALUTs 5.84 ns	10770 ALUTs 6.55 ns
DMFFT	4466 ALUTs 4.86 ns	5365 ALUTs 4.90 ns	6819 ALUTs 5.50 ns	8564 ALUTs 5.90 ns	10389 ALUTs 6.58 ns
Memory saving	33 %	31 %	27 %	24 %	21.9 %
ALUT's gain	16.2 %	11.7%	9.29 %	4.68 %	3.5%
η's gain	24 %	15.1 %	9.4%	4.2 %	3.54 %

6.2 Application de l'approche de conception dans le contexte DMFFT

De l'étude de l'opérateur commun du paragraphe précédent, nous déduisons qu'il est possible de mettre en oeuvre l'opérateur DMFFT à la fois pour effectuer tout calcul de FFT et le décodage RS dans $GF(F_t)$ ses deux algorithmes les plus complexes (calcul du syndrome er recherche de Chien). C'est ce sur quoi le graphe de la Fig. 31 insiste en prenant le cas

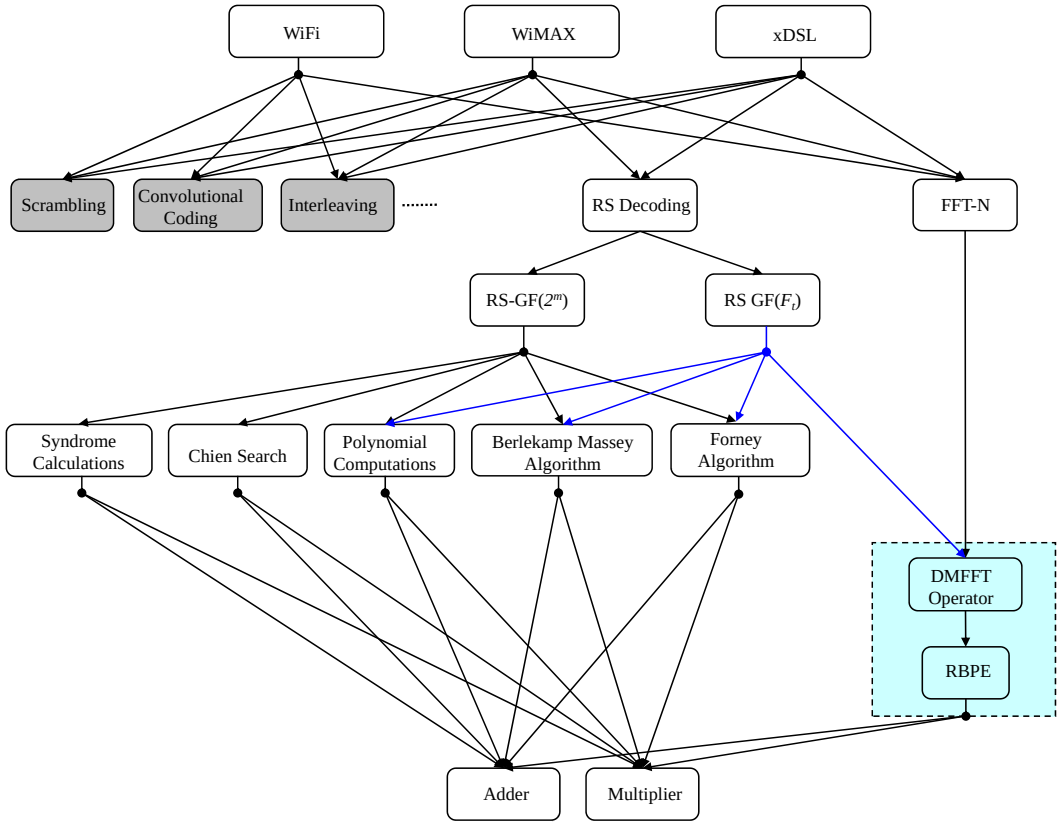

Figure 31: Graphe d'un équipement tri-standards en utilisant $GF(F_t)$ pour le codage RS

imaginaire où les trois standards WiFi, WiMAX et xDSL auraient utilisé ce codage RS à la place de celui dans $GF(2^m)$. La décomposition de l'algorithme de décodage RS n'est pas détaillée dans ce résumé hormis dans la Fig. 29, mais elle peut être trouvée dans le texte anglais complet. Précisons juste que la DMFFT peut être notamment utilisée pour les 2 étapes suivantes du décodage RS : le calcul du syndrome et la recherche de Chien.

Les noeuds à gauche en gris ont été représentés pour illustrer ce que devrait intégrer un graphe complet, au-delà de l'étude de la DMFFT. La Fig. 32 zoome sur la partie basse du graphe.

Les paramètres associés aux arcs sont les nombres d'appels (NoC), ceux représentés ici sur les noeuds sont les coûts de fabrication (BC) en termes de nombre d'ALUT, et le coût d'exécution est exprimé en nanosecondes. On remarque qu'il est possible de paramétrer certains poids associés aux arcs, comme c'est le cas entre le noeud DMFFT-N et le noeud RBPE suivant la taille de la FFT. On retrouve dans la Table 4 les valeurs des coûts pour la DMFFT et les noeuds sous-jacents.

Prenons l'exemple du cas du décodage RS pour détailler les résultats du calcul d'optimisation. La Table 5 donne les coûts en termes d'additions et de multiplications pour différentes fonc-

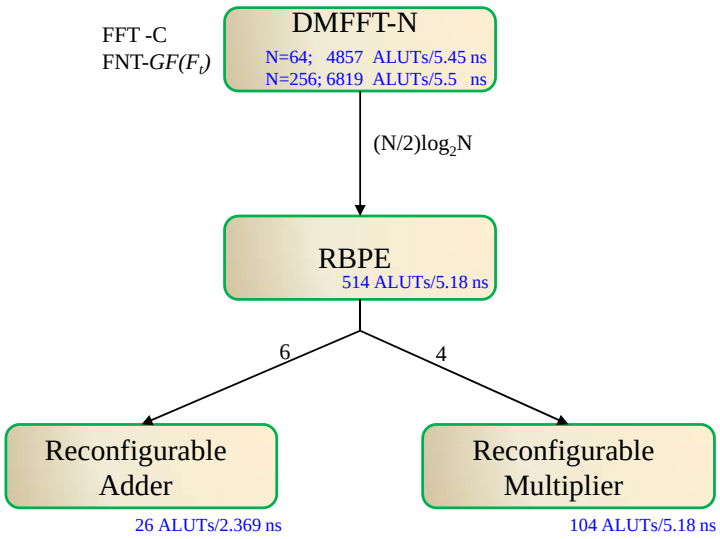

Figure 32: Décomposition possible de la DMFFT

Table 4: Nombre d'appels du bloc DMFFT

Block Name	**No of ALUT**	**Execution Time(ns)**	**No of Calls**
DMFFT-256	6819	5.5	1
RBPE	514	5.18	128= $(N/2)$
Re-Adder	26	2.37	6
Re-Multiplier	104	5.18	4

tions dans le cas d'un décodage RS classique.

Le coût du noeud de décodage RS peut être obtenu par l'équation (7) :

$$Cost_{level} = \Big(\sum_{i=1}^{i=m} CC_i NoC_i \Big) \times NoC_{i+1} + \sum_{i=1}^{i=m} BC_i.N_i \tag{24}$$

Comme nous ne considérons que le noeud de décodage RS, alors $NoC_{i+1} = 1$ et l'équation (24) devient :

Table 5: Nombre d'appels en termes de multiplications et additions pour le décodage RS classique pour t=8 et N=256

No. of Calls	**Additions**	**Multiplications**
Syndrome Comp.	4096	4096
Chien Search	2048	2048
Polynomial Comp.	384	392
Berlekamp Massey	512	512
Forney Algorithm	8	16

$$Cost_{RSDecoding} = \Big(\sum_{i=1}^{i=5} CC_i NoC_i \Big) \times NoC_{i+1} + \sum_{i=1}^{i=5} BC_i.N_i \tag{25}$$

En utilisant le première option pour concevoir le décodage RS, alors :

$$\begin{aligned} Cost_{RS-GF(2^m)} &= Cost_{Syndrome} + Cost_{Chien} + Cost_{Poly} \\ &+ Cost_{Berlekamp} + Cost_{Forney} \end{aligned} \tag{26}$$

soit,

$$\begin{aligned} Cost_{RS-GF(2^m)} &= \big((4096 + 2048 + 392 + 512 + 16) \times CC_{Multiplier} \\ &+ (4096 + 2048 + 384 + 512 + 8) \times CC_{Adder}\big) \times 1 \\ &+ BC_{Multiplier} + BC_{Adder} \end{aligned} \tag{27}$$

et donc,

$$Cost_{RS-GF(2^m)} = \big(7064 \times 5.18 + 7048 \times 2.37\big) \times 1 + 104 + 26 = 53425.28 \tag{28}$$

Alors qu'en utilisant la seconde pour réaliser le décodage RS, l'équation (25) devient:

$$Cost_{RS-GF(F_t)} = Cost_{Poly} + Cost_{Berlekamp} + Cost_{Forney} + Cost_{DMFFT} \tag{29}$$

soit,

$$\begin{aligned} Cost_{RS-GF(F_t)} &= \big((392 + 512 + 16) \times CC_{Multiplier} \\ &+ (384 + 512 + 8) \times CC_{Adder} + 2 \times CC_{DMFFT}\big) \times 1 \\ &+ BC_{Multiplier} + BC_{Adder} + BC_{DMFFT} \end{aligned} \tag{30}$$

et donc,

$$Cost_{RS-GF(F_t)} = \big(920 \times 5.18 + 904 \times 2.37 + 2 \times 5.5\big) \times 1 + 104 + 26 + 6819 = 13868.08 \tag{31}$$

On obtient donc la fonction de coût suivante pour le cas d'un graphe réduit au noeud de décodage RS :

$$\mathbf{C_{RSDecoding}} = \min\big(Cost_{RS-GF(2^m)}, Cost_{RS-GF(F_t)}\big) = \mathbf{Cost_{RS-GF(F_t)}} = 13868.08$$

Cela montre que le coût du décodage RS est bien moindre dans GF(F_t) grâce à l'utilisation de l'opérateur DMFFT.

Poussons plus loin l'exploration dans la branche de décodage RS avec GF(F_t), afin de voir si l'opérateur DMFFT peut être un candidat en tant qu'opérateur commun.

Notons que le coût de la DMFFT seule est :

$$\begin{aligned} Cost_{DMFFT} &= BC_{DMFFT} + CC_{DMFFT} \\ &= 6819 + 5.5 = 6824.5 \end{aligned} \tag{32}$$

Si maintenant nous utilisons l'opérateur RBPE pour la mettre en oeuvre, le coût de la DMFFT devient en utilisant l'équation (7) :

$$\begin{aligned} Cost_{DMFFT_{RBPE}} &= CC_{RBPE} \times NoC + BC_{RBPE} \\ &= 5.18 \times 128 + 514 \\ &= 1177.04 \end{aligned} \tag{33}$$

En descendant encore dans le graphe, le RBPE peut être constitué par un additionneur et un multiplieur. Le coût de la DMFFT utilisant un additionneur et un multiplieur est donc de :

$$Cost_{DMFFT_{Adder+Multiplier}} = \Big(\sum_{i=1}^{i=2} CC_i NoC_i \Big) \times NoC_{i+1} + \sum_{i=1}^{i=2} BC_i \tag{34}$$

soit,

$$\begin{aligned} Cost_{DMFFT_{Adder+Multiplier}} &= \Big(CC_{Adder} \times NoC_{Adder} + \\ &\quad CC_{Multiplier} \times NoC_{Multiplier} \Big) \times 128 + \\ &\quad BC_{Adder} + BC_{Multiplier} \end{aligned} \tag{35}$$

$$\tag{36}$$

et donc,

$$\begin{aligned} Cost_{DMFFT_{Adder+Multiplier}} &= \Big(2.369 \times 6 + 5.18 \times 4 \Big) \times 128 + \\ &\quad 26 + 104 = 4601.55 \end{aligned} \tag{37}$$

$$\tag{38}$$

Cela montre que réaliser la DMFFT avec le RBPE donne un coût minimal comme :

$$Cost_{DMFFT_{RPBE}} < Cost_{DMFFT_{Adder+Multiplier}} < Cost_{DMFFT} \tag{39}$$

En d'autres termes, le résultat de l'équation (31) en utilisant maintenant l'opérateur RBPE pour la DMFFT devient :

$$Cost_{RS-GF(F_t)} = 8876.16 \tag{40}$$

soit,

$$\mathbf{C_{RSDecoding}} = \mathbf{Cost_{RS-GF(F_t)}} = 8876.16 \tag{41}$$

Cela montre que la mise en oeuvre de la DMFFT via l'opérateur RBPE permet une réduction de coût. C'est donc le noeud RBPE qui donne la solution au coût minimal pour le décodage RS.

6.3 Résultats et conclusions

Le cas de l'utilisation de la FFT pour le codage de canal de type RS a été exploré dans ce chapitre. Un opérateur DMFFT a été proposé comme opérateur commun par [43]. L'implantation sur FPGA a permis d'en mesurer le coût de fabrication et d'exécution et ainsi de pouvoir l'insérer dans la méthode proposée dans ce manuscrit. Ce travail a donné lieu à une publication revue [45].

Par ailleurs, afin de voir la portée que peut avoir la méthode proposée, il est démontré dans cet exemple que si l'approche par opérateur commun pour ses avantages en terme de reconfiguration et de complexité avait été pris en considération comme ici (contexte possible d'équipements multi-standards) dans les standardisations WiFi, WiMAX et xDSL, alors le codage de canal RS aurait été choisi dans $GF(F_t)$ au lieu de $GF(2^m)$.

7 Cas d'étude : opérateur commun LFSR dans un équipement de radio logicielle multi-standards

Ce chapitre 7 étudie particulièrement le cas de l'opérateur LFSR (*Linear feedback shift register*). Quatre architectures communes sont développées afin de concevoir 16 opérateurs communs LFSR qui permettent de supporter plusieurs opérations d'un équipement de radio intelligente suivant la manière dont ils sont combinés. Les détails d'implantation sont donnés dans le cas d'un FPGA Cyclone-II d'Altera.

7.1 Opérateur LFSR

L'annexe E rappelle les fondements et les architectures de l'opérateur LFSR. Quatre architectures ont été proposées par Alaus dans [46], basées sur les deux familles de LFSR, *Fibonacci* ou *Galois*, et sur le fait qu'elles sont reconfigurables :

- RG-LFSR (*Reconfigurable Galois LFSR*),
- RF-LFSR (*Reconfigurable Fibonacci LFSR*),
- R-LFSR (*Reconfigurable LFSR*),
- ER-LFSR (*Extended Reconfigurable LFSR*).

Un LFSR est un registre à décalage (*shift register*) dont le bit d'entrée est une combinaison linéaire des contenus précédents de ses registres. Ceci présente la particularité que si l'on veut faire appel à une sous-structure du LFSR comme opérateur commun et ordonnancer ses appels ensuite, il faut garder en mémoire les valeurs de tous les registres, les indices des rebouclages, les entrées et les sorties, ce qui implique un surcoût de mémoire (et sa gestion) important. D'autre part, même si des structures sont remplaçables,

Table 6: Décomposition en LFSR Indépendants

LFSR common operators	**Logic cells**
RG-LFSR4	5
RG-LFSR8	10
RF-LFSR4	5
RF-LFSR8	10
R-LFSR4	6
R-LFSR8	10
ER-LFSR4	11
ER-LFSR8	16

disons par le ER-LFSR (dont la structure est exposée en annexe E), elles peuvent avoir différentes configurations. Par exemple, dans ce chapitre, l'opérateur commun requis doit avoir une chaîne LFSR de 32 registres, deux chaînes de 27 registres ou trois chaînes de 8 registres. Ainsi, la conception et la mise en place d'un seul opérateur commun basé sur l'architecture ER-LFSR présente une difficulté certaine.

Afin de s'affranchir des problèmes d'ordonnancement, nous avons choisi de masquer le problème en créant une banque d'opérateurs communs (COB - *Common Operator Bank*) où de petits opérateurs ER-LFSR de taille identique sont interconnectés. Ces connections sont paramétrables et permettent de mettre en oeuvre des chaînes ER-LFSR de tailles différentes. Le but est d'éviter les problèmes liés à l'ordonnancement en implantant autant d'opérateurs que requis par le standard le plus exigeant.

Prendre ainsi le pire cas peut paraître en contradiction avec la philosophie des opérateur communs car l'approche par opérateurs communs approche tend à trouver la plus petite entité commune (en fait la moins coûteuse, ce qui peut être un peu différent de la plus petite). De plus, l'approche par le pire cas peut paraître évidente, mais elle a l'avantage d'offrir déjà une première économie par rapport au cas velcro (où le coût multi-standards est la somme des coûts de chacun des standards), ce qui est cohérent avec les buts de l'approche par opérateurs communs. Cependant, nous verrons par la suite que c'est l'opérateur commun étant capable d'effectuer le pire cas (ce qui signifie automatiquement tous les autres qui sont nécessaires dans l'équipement aussi - voir Table 7) que nous choisirons, et nous pas la structure entière du pire cas.

Ainsi, la Fig. 33 illustre plusieurs cas d'allocations possibles. Avec le COB il est possible de replacer exactement la structure dont on a besoin. En fonction des structures les plus petites à mettre en oeuvre nous avons choisi des tailles de 4 et 8 comme il est expliqué ci-après.

7.2 LFSR et approche de conception par graphe

Dans le cas d'un FPGA ALTERA Cyclone-II, la Table 6 donne le nombre de cellules logiques nécessaires pour la réalisation de chaque catégorie de LFSR étudiée.

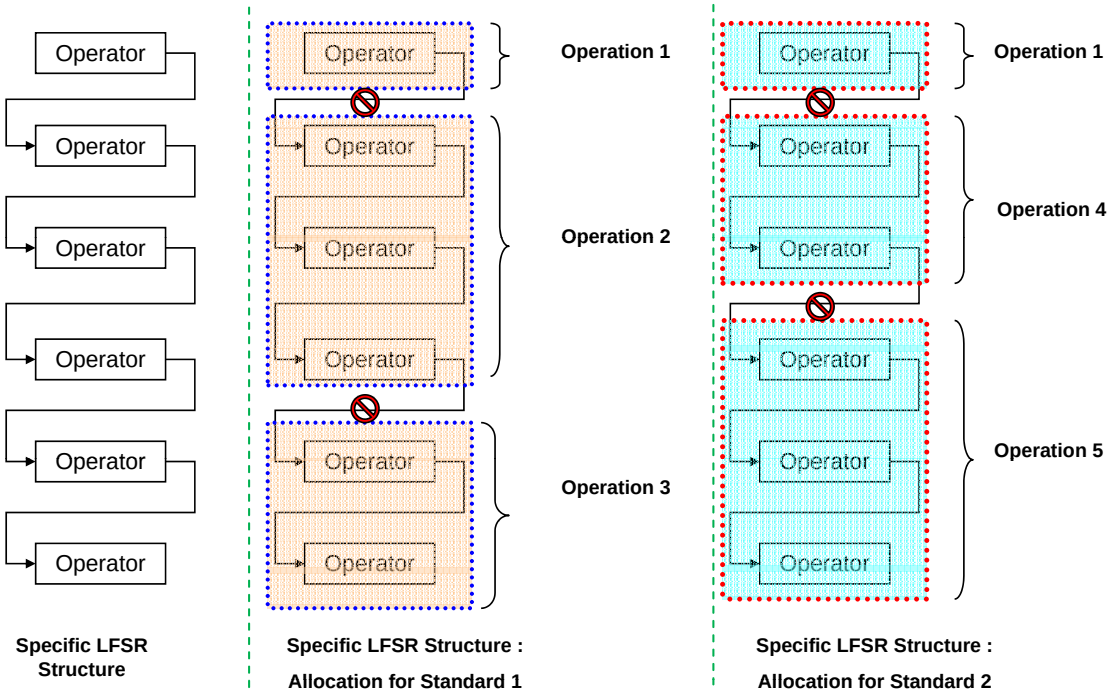

Figure 33: Banque d'opérateurs communs (COB)

La Table 7 liste ensuite le nombre de LFSR nécessaires pour mettre en oeuvre diverses fonctions de plusieurs standard, WiFi, WiMAX et LTE. Dans cette table RG-4 signifie un opérateur commun LFSR constitué de 4 unités RG-LFSR. On remarque qu'avec les LFSR-4 et LFSR-8 (des quatre cas: RG, RF, R et ER), on peut effectuer tous les cas de la Table 6. Les tailles 4 et 8 ont été choisies de la manière suivante [46]. Toutes les structures de LFSR de taille allant de 2 à 30 ont été évaluées en termes de complexité en nombre de transistors. Les LFSRs de taille 4 ont le plus faible complexité. En outre, la taille 4 est la plus petite taille requise dans tous les cas de scénarios de conception envisagés. Les LFSRs de taille 8 sont aussi considérés car les structures sur 8 bits sont les plus courantes.

En combinant les tables 6 et 7, on peut aisément déduire la complexité de ces fonctione en termes de nombre de cellules logiques. Prenons l'exemple de WiFi. Trois fonctions peuvent être réalisées en utilisant l'opérateur LFSR : *scrambling*, *CRC* et le codage convolutif (*CC*). Afin d'implanter le scrambling les 8 opérateurs proposés LFSR peuvent convenir, mais pour le CRC, RG-LFSR4 et RG-LFSR8 ne conviennent pas. Ou nous utilisons n'importe lequel des 6 opérateurs restants pour le CRC et le scrambling i.e. RG-LFSR4 (4 RG-LFSR4 pour le scrambling + 8 RG-LFSR4 pour le CRC) ou nous utilisons une combinaison d'opérateurs LFSR i.e. par exemple 4 fois un RF-LFSR4 pour le scrambling et 4 fois un RG-LFSR8 pour le CRC. Il est à noter que deux choix de conception sont possibles, soit un COB homogène constitué d'éléments d'un seul type, soit un COB hétérogène.

En procédant ainsi de suite pour les 3 standards envisagés, il est possible de mettre en oeuvre toutes les configurations de CRC avec le R-LFSR4. Il en faut 8 pour WiFi, 16 pour WiMAX (afin de supporter les deux cas CRC16 et CRC32) et finalement 12 pour LTE

Table 7: Opérations de WiFi/WiMAX/3GPP-LTE et nombre de LFSRs

Function	**RG-LFSR4**	**RG-LFSR8**	**RF-LFSR4**	**RF-LFSR8**	**R-LFSR4**	**R-RLFR8**	**ER-LFSR4**	**ER-LFSR8**
WiFi-Complexity in terms of number of LFSRs								
Scrambler	4	2	4	2	4	2	4	2
CRC16	-	-	8	4	8	4	8	4
CC-802.11b	-	-	-	-	4	2	4	2
CC-802.11g	-	-	-	-	-	-	6	6
IEEE 802.16 d/e (WiMAX) Complexity in terms of number of LFSRs								
Scrambler-15	8	4	8	4	8	4	8	4
+*Scrambler-11*	6	4	6	4	6	4	6	4
or +Scrambler-22	12	6	12	6	12	6	12	6
CRC32	-	-	16	8	16	8	16	8
CRC16	-	-	8	4	8	4	8	4
CC-1/2	-	-	-	-	4	2	4	2
TC-1/2	-	-	-	-	-	-	1	1
TC-1/3	-	-	-	-	-	-	2	2
3GPP LTE Complexity in terms of number of LFSRs								
Scrambler-UL	8	4	8	4	8	4	8	4
+*Scrambler-DL*	6	4	6	4	6	4	6	4
CRC32	-	-	12	6	12	6	12	6
CRC24	-	-	8	4	8	4	8	4
CRC16	-	-	6	4	6	4	6	4
CRC8	-	-	-	-	4	2	4	2
CC-1/2	-	-	-	-	4	2	4	2
CC-1/3	-	-	-	-	6	3	6	3
TC	-	-	-	-	2	2	2	2

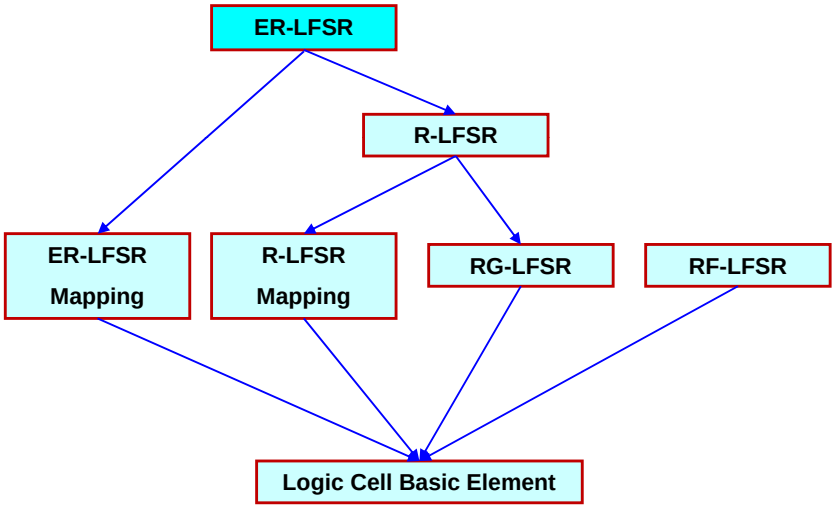

Figure 34: LFSRs Combinés

(pour supporter les CRC8, CRC16, CRC24 et CRC32). On en déduit que 16 R-LFSR4 permettent d'assurer l'exécution de tous les cas de CRC exigés dans les 3 standards considérés.

La mise en oeuvre de cette approche intuitive se révèle finalement contre-productive puisque dans le cas d'un équipement tri-standards avec 2 modes LTE et un mode WiFi, une telle implantation présente un surnombre de 5 % en termes de nombre de cellules logiques par rapport à une approche Velcro. En effet, ce sont des COB utilisant des ER-LFSR8 qui sont utilisés, ce qui représente 352 cellules logiques contre 336 pour l'approche Velcro (obtenus après synthèse).

Il est bien sûr possible également de combiner plusieurs jeux de LFSR afin d'obtenir de meilleurs résultats. Lançons le processus d'optimization pour explorer les solutions. La figure 34 explique que les différentes structures de LFSR peuvent se combiner entre elles en ajoutant des portes logiques de liaison ou *mapping*.

On obtient alors en terme de complexité la nouvelle Table 8. Avec une telle décomposition, on peut remarquer que les (E)R-LFSR sont la combinaison de R(G)-LFSR avec d'autres structures basées sur des éléments logiques :

- R-LFSR = RG-LFSR + 1.Logic Cells (R-LFSR Mapping)
- R-LFSR4 = RG-LFSR4 + 5.Logic Cells (R-LFSR4 Mapping)
- R-LFSR8 = RG-LFSR8 + 9.Logic Cells (R-LFSR8 Mapping)

et

- ER-LFSR = R-LFSR + Logic Cells (ER-LFSR Mapping) = RG-LFSR + 1.Logic Cells

Table 8: LFSRs combinés

Combined LFSR common operators	**Logic cells**
RG-LFSR4	5
RG-LFSR8	10
RF-LFSR4	6
RF-LFSR8	11
R-LFSR4	11
R-LFSR8	20
ER-LFSR4	17
ER-LFSR8	27

- ER-LFSR4 = R-LFSR4 + 6.Logic Cells (ER-LFSR4 Mapping)= RG-LFSR4 + 11.Logic Cells.
- ER-LFSR8 = R-LFSR8 + 7.Logic Cells (ER-LFSR8 Mapping) = RG-LFSR8+ 16.Logic Cells.

Remarquez que les relations ci-dessus expriment juste le principe de combinaison (structurelle) des opérateurs et ne doivent pas être considérées pour calculer le coût équivalent. Par conséquent, tous les opérateurs LFSR combinés peuvent avoir plus d'un seul coût, en fonction du type de LFSR de plus bas niveau choisi pour être combinés ensemble.

La section suivante montre comment le processus d'optimisation peut aider à réduire le précédent coût en utilsant des LFSR combinés.

7.3 Conception d'un équipement tri-standards

Revenons au problème de conception multi-standard dans le cas d'un équipement tri-standards LTE1, LTE2 et WiFi. La Fig. 35 restreint le graphe aux fonctions pouvant utiliser l'opérateur LFSR. TBxx signifie blocks transparents. Ils ne sont là que pour alléger visuellement le graphe. La décomposition des opérateurs LFSRs n'est pour l'instant pas utilisée mais le sera ensuite. Les opérateurs LFSR en bas de la Fig. 35 sont précédés d'un nombre qui indique le facteur de duplications de l'opérateur, i.e. que 22ER signifie un ER-LFSR4/8 dupliqué 22 fois.

Le but est de trouver la meilleure combinaison d'opérateurs donnant un coût minimal. Pour cela le coût de chaque noeud est calculé. Le coût BC est directement calculé depuis le nombre de LFSRs requis et du nombre de cellules logiques requis pour mettre en oeuvre un LFSR. Pour le coût CC, comme le problème d'ordonnancement a été masqué par l'utilisation des COBs, nous le fixons à 1. Nous pouvons considérer en effet qu'ils sont exécutés en 1 coup d'horloge. Comme par ailleurs chaque opérateur ou combinaison d'opérateurs est appelé en une fois pour faire sa tache, le NoC est égal à 1.

Rappelons que dans le cas de la Fig. 35, les opérateurs LFSR ne sont pas combinés. En appliquant alors le programme d'optimisation sur le graphe de la Fig. 35 on obtient finalement l'équation :

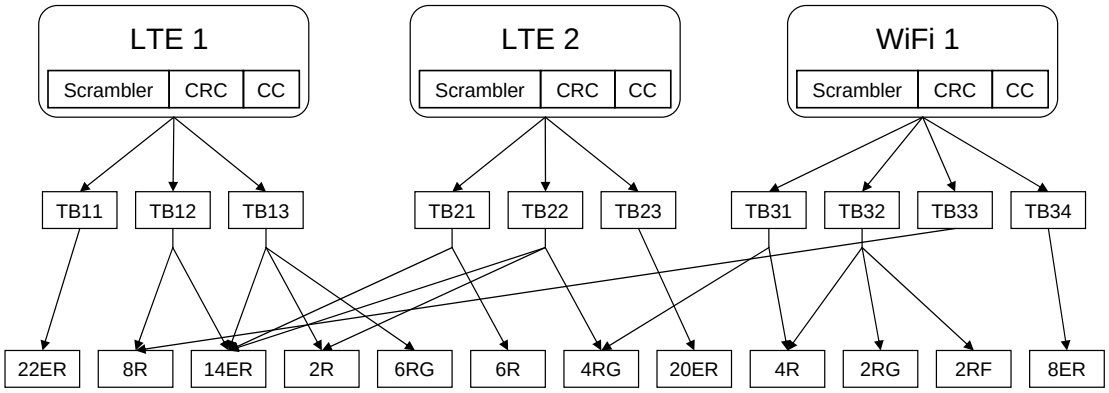

Figure 35: Equipement tri-standards avec l'opérateur commun LFSR

$$\begin{aligned} \mathbf{C_{LTE1,LTE2,WiFi1}} &= Cost_{14ER+8R+2R+4RG} \\ &= 14 \times 16 + 8 \times 10 + 2 \times 10 + 4 \times 10 \\ &= 364 \end{aligned} \tag{42}$$

Il est à noter que les coûts des LFSR-8 sont ici utilisés pour les raisons évoquées dans paragraph 7.2. L'optimisation donne donc dans ce cas les noeuds *14ER, 8R, 2R and 4RG* afin d'obtenir le coût minimal pour la mise en oeuvre de l'équipement tri-standards.

Si maintenant nous compliquons l'exemple en introduisant la décomposition de l'opérateur 22ER de la Fig. 36 comme évoqué précédemment. L'équation (42) devient alors :

$$\begin{aligned} \mathbf{C_{LTE1,LTE2,WiFi1}} &= Cost_{14ER+2R+4RG} \\ &= 14 \times 16 + 2 \times 10 + 4 \times 10 \\ &= 284 \end{aligned} \tag{43}$$

La nouvelle optimisation sélectionne les opérateurs *14ER, 2R* and *4RG* afin d'obtenir le coût minimal pour la mise en oeuvre de l'équipement tri-standards. Les opérateurs *14ER* et *2RG* sont communs aux deux solutions. En revanche, l'opérateur *8R* du cas précédent est remplacé par l'opérateur *2R* qui peut à la fois mettre en oeuvre le R-LFSR4 et le R-LFSR8.

Il est à noter que l'impact de l'utilisation d'opérateurs communs est d'autant plus important que le nombre de standards considérés est grand. Dans le cas simple de 3 standards ici, le nombre de cellules logiques passe de 336 (obtenus par synthèse) dans le cas velcro à la combinaison de 14 ER, 2R et 4 RG soit 284 cellules logiques (équation (43)).

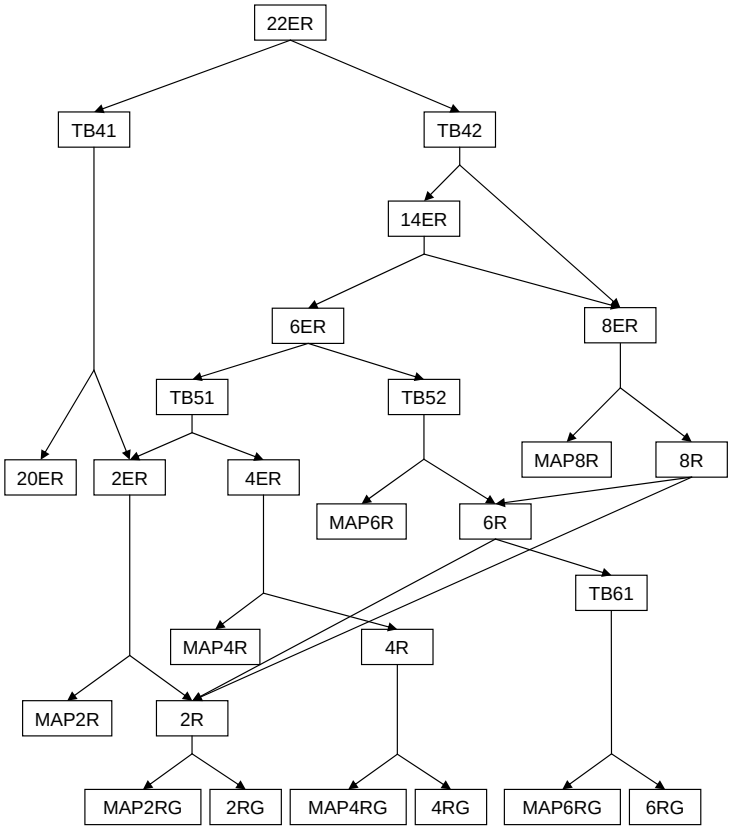

Figure 36: Décomposition de l'opérateur 22ER en opérateurs plus petits

7.4 Résultats et conclusions

Cette étude montre l'intérêt à considérer l'opérateur LFSR comme un opérateur commun afin d'être utilisé pour de nombreuses fonctions tels que *scrambling*, le CRC et le codage convolutif. Nous nous sommes appuyés pour cela sur des chiffres d'implantation sur cible FPGA Cyclone-II de Altera afin d'avoir des résultats réalistes. Ce travail a donné lieu à une publication revue [47].

8 Cas d'étude : opérateur commun FRMFB

Un élément clef d'un équipement multi-standards est l'organe qui est capable de "canaliser" le signal reçu (on parle de *channelizers* en anglais ou canaliseurs), autrement dit d'extraire un canal (fréquentiel) correspondant à un standard. En effet, idéalement, une numérisation à large bande de plusieurs canaux est effectuée en tête de réception dans la philosophie radio logicielle. Dans le chapitre 8, plusieurs propositions de bancs de filtres sont détaillées [48] et intégrée dans un graphe afin de lancer la méthode proposée dans cette thèse.

8.1 Bancs de filtres

La solution classique de l'extraction canal par canal (*PC - Per Channel*) par des filtres figés n'est plus rentable dans le cas où le nombre de canaux à extraire en réception augmente. Et cela s'avère d'autant plus vrai s'il est nécessaire d'extraire plusieurs canaux à la fois tel que dans une station de base [49].

La solution passe alors par les bancs de filtres. Les plus connus sont les bancs de filtres basés sur la transformée de Fourier discrète (DFTFB pour *Discrete Fourier Transform Filter Bank*) [49, 50]. Au lieu de N filtres, seul un filtre passe bas suivi par une FFT est nécessaire. Cependant, cela implique des contraintes et notamment que les bandes à extraire soient identiques.

Une alternative a été aussi proposée dans [49]: GFB (*Goerzel Filter Bank*). Cette solution résoud les problèmes liés aux contraintes de localisation en fréquence et de sélectivité du filtre prototype qui existent dans le cas DFTFB. Notons aussi que [51] propose un banc de filtres basé sur la combinaison de filtres polyphase et de DFT.

Une quatrième limitation n'est pas encore résolue dans ces 3 cas. Il s'agit de celle qui permet de reconfigurer le même banc de filtres pour s'adapter à différents standards. La version anglaise de la thèse donne des exemples de canaliseurs issus de la littérature. Citons seulement [48] qui propose un banc de filtres reconfigurable basé sur les techniques de masquage de la réponse en fréquence (*Frequency Response Masking Filter Bank* or FRMFB). Les avantages supplémentaires par rapport aux autres solutions sont que la reconfigurabilité est désormais possible au niveau du filtre mais aussi de sa structure, que la vitesse de filtrage est améliorée et que la complexité en est réduite [48].

La méthode *frequency masking* a été introduite en 1986 dans [52] afin de mettre en oeuvre des filtre FIR très sélectifs d'ordre réduit. L'architecture d'un tel filtre est proposée dans la Fig. 37. Elle est constituée de :

- Filtre prototype filter: $H_1(z^M)$
- Filtre complémentaire à $H_1(z)$: $z^{M(N_a-1)/2}$
- Filtre de masquage haut : $H_{FMA}(z)$
- Filtre de masquage bas : $H_{FMC}(z)$

8.2 Graphe du canaliseur multi-standards

Si l'on restreint l'étude de conception d'un équipement au canaliseur, on peut retrouver le graphe de la Fig. 38. Dans cet exemple, nous considérons le coût des noeuds en termes de nombres d'additions et de soustractions.

8.3 Optimisation du graphe canaliseur

L'étude porte sur la conception d'un équipement bi-standards GSM, WCDMA dont on veut extraire 5 canaux pour chaque standard. Les canaux sont de 200 kHz pour le GSM

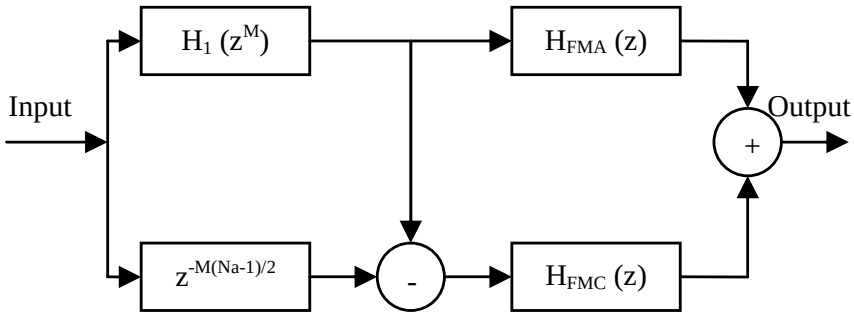

Figure 37: FIR filter architecture based on FRM technique

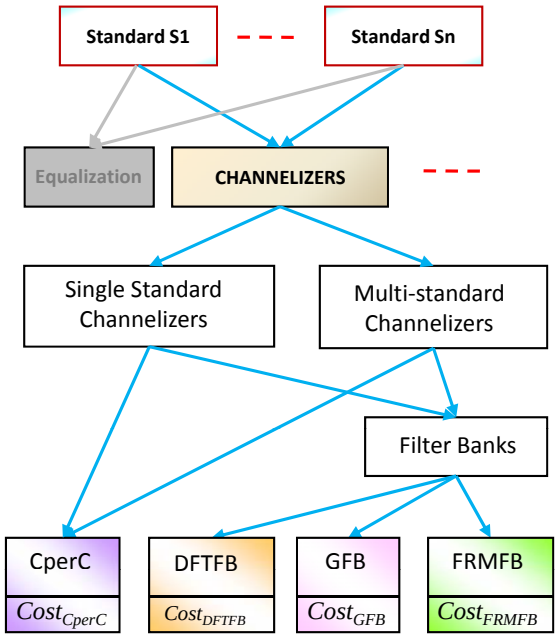

Figure 38: Canaliseurs pour un équipement SDR multi-standards

et de 3.86 MHz pour l'UMTS.

L'étude est restreinte aux possibilités dont on a les chiffres de complexité PC, DFTFB et FRMFB. Le processus d'optimisation consiste à trouver la solution à l'équation :

$$\mathbf{C_{Channelizer}} = \min\left(Cost_{CPerC}, Cost_{DFTFB}, Cost_{GFB}, Cost_{FRMFB}\right) \quad (44)$$

Si nous expliquons plus en détail le coût du noeud FRMFB à l'aide de l'équation (7) rappelée ci-dessous :

Table 9: Complexité du FRMFB pour l'extraction de 5 canaux GSM et 5 canaux WCDMA

Name of Filter	**No. of Taps**	**No. of Adders**	**No. of Multipliers**
Modal Filter	81	41	80
FMA_GSM	600	$300 \times 5 = 1500$	$599 \times 5 = 2995$
FMA_WCDMA	250	$125 \times 5 = 625$	$249 \times 5 = 1245$
FMC_WCDMA	250	$125 \times 5 = 625$	$249 \times 5 = 1245$
Total complexity	-	2791	5565
Complexity of PC/ DFTFB	1400	7000	13990

$$Cost_{level} = \Big(\sum_{i=1}^{i=m} CC_i NoC_i \Big) \times NoC_{i+1} + \sum_{i=1}^{i=m} BC_i.N_i \tag{45}$$

Si l'on considère le noeud FRMFB, $NoC_{i+1} = 1$ et (45) devient :

$$Cost_{FRMFB} = \Big(\sum_{i=1}^{i=4} CC_i NoC_i \Big) \times 1 + \sum_{i=1}^{i=4} BC_i.N_i \tag{46}$$

So

$$\begin{aligned} Cost_{FRMFB} &= Cost_{ModalFilter} + Cost_{FMA_{GSM}} + Cost_{FMA_{WCDMA}} \\ &+ Cost_{FMC_{WCDMA}} \end{aligned} \tag{47}$$

La Table 9 donne les complexités des différents éléments des FRMFB.
On en déduit que :

$$\begin{aligned} Cost_{FRMFB} &= \big((41 + 1500 + 625 + 625) \times CC_{Adder} \\ &+ (80 + 2995 + 1245 + 1245+) \times CC_{Multiplier}\big) \\ &+ BC_{Adder} + BC_{Multiplier} \end{aligned} \tag{48}$$

Considérant une implantation du FRMFB sur un composant FPGA Stratix-II de Altera, on peut reprendre les BC et CC du multiplieur et de l'additionneur dans la table 4, et alors :

$$Cost_{FRMFB} = \big(2791 \times 2.37 + 5565 \times 5.18\big) \times 1 + 26 + 104 = 35571.37 \tag{49}$$

De la même manière, le coût du canaliseur PC ou DFTFB se calcule ainsi car il n'y a pas vraiment de différence de complexité pour ces 2 cas dans le contexte considéré ici par rapport à celle du FRMFB :

$$\begin{aligned} Cost_{PC/DFTFB} &= \big(NoC_{Adder} \times CC_{Adder} + NoC_{Multiplier} \times CC_{Multiplier}\big) \\ &+ BC_{Adder} + BC_{Multiplier} \end{aligned} \tag{50}$$

soit

$$\begin{aligned} Cost_{PC/DFTFB} &= \big((7000) \times CC_{Adder} + (13990) \times CC_{Multiplier}\big) \\ &+ BC_{Adder} + BC_{Multiplier} \end{aligned} \tag{51}$$

et donc

$$Cost_{PC/DFTFB} = \big(7000 \times 2.37 + 13990 \times 5.18\big) + 26 + 104 = 89188.20 \tag{52}$$

Le FRMFB est donc l'opérateur commun de la Fig. 38.

8.4 Résultats et conclusion

Le FRMFB est choisi comme candidat dans ce cas d'étude de la canalisation pris seul pour un équipement SDR multi-standards. Cependant, si nous ajoutions le standard WiFi au cas d'étude et que nous considérions aussi la FFT nécessaire pour effectuer la modulation/démodulation OFDM de WiFi, alors le résultat pourrait être différent. Car alors, la FFT pourrait être utilisée à la fois pour la modulation/démodulation et pour la canalisation. C'est l'intérêt de la méthode de conception proposée ici : éviter de choisir des optimisations locales et permettre d'envisager des optimisations globales à l'ensemble de la chaîne de traitement. C'est ce que laissait entrevoir la Fig. 7 du chapitre 2 et que nous allons étudier dans le chapitre 9.

9 Cas d'étude d'un graphe complexe

Nous combinons dans ce chapitre 9 deux des cas étudiés dans la section III de ce document afin d'effectuer un premier pas (à bien sûr poursuivre dans une conception complète) du cas de conception d'un équipement multi-standards SDR. La cas d'étude vise à regrouper dans un même graphe les opérateurs DMFFT du chapitre 6 et FRMFB du chapitre 8 pour la conception d'un équipement multi-standards WiFi/WiMAX/UMTS. Le but est de trouver un optimum global dans ce cas, qui peut se distinguer des optimums locaux trouvé dans chacun des cas (ce sur quoi portent habituellement les travaux d'optimisation architecturale). Ce dernier exemple a pour but de véritablement mettre en évidence l'avantage de l'approche préconisé dans ce travail de thèse.

9.1 Opérateurs communs

La Fig. 39 reprend les opérateurs DMFFT du chapitre 6 et FRMFB du chapitre 8. Le but est d'essayer de voir s'il y a des possibilités de ré-utilisation des opérateurs communs trouvés pour l'un dans le cas de l'autre. Ainsi, on pressent que l'opérateur DMFFT pourraient aussi être utilisé dans les bancs de filtre, en plus des cas déjà identifiés de la modulation/démodulation OFDM et du codage/décodage RS. Cela pourrait contredire la conclusion (locale) du chapitre 8 qui a fait le choix FRMFB au profit de la DFTFB qui pourrait alors effectuer sa transformée de Fourier avec l'opérateur DMFFT. C'est tout l'intérêt du travail de cette thèse qui consiste à fournir une méthode et un outillage

pour trouver un optimum global à la conception, au détriment d'a priori sur les optimums locaux parfois préconisés. La condition est de disposer de graphes suffisamment complétés.

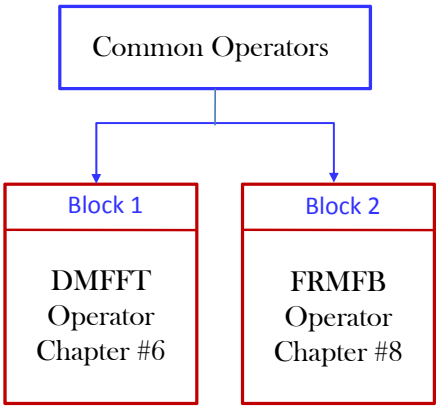

Figure 39: An generalized view of common operators

9.2 Graphe de conception

Le graphe du système tri-standards étudié est proposé dans la Fig. 40. Comme dans les études précédentes, il ne comporte pas toutes les opérations des couches protocolaires complètes des trois standards (ce qui aurait été impossible à obtenir lors de la thèse et qui ne faciliterait pas les explications voulues ici), mais il est réduit aux fonctions de démodulation OFDM, de décodage RS et de canalisation. Ainsi, nous pouvons espérer combiner les résultats obtenus pour les opérateurs DMFFT et FRMFB. L'interface graphique développée dans ces travaux est utilisée sur la Fig. 40 pour représenter le graphe et associer les paramètres de coût à chaque entité (arc ou noeud).

Afin d'améliorer la lisibilité, le graphe a été repris dans la Fig. 41.

9.3 Coûts

Les coûts sont extraits des tableaux de complexité de l'Annexe B (de la version anglaise). Mais ils ne sont pas dans la même unité, ceux de la DMFFT étant en nombre d'ALUTs et ceux du LFSR étant en nombre de cellules logiques (LC) de multiplieurs et d'additionneurs.

Les paramètres de coût des différents noeuds de la Fig. 41 sont donnés en Table 10.

Certains de ces coûts sont extraits de la Table 4. Les coûts BC/CC pour les FRMFB et DFTFB sont issus de [48] en nombre de portes. Ils sont alors convertis en nombre d'ALUTs avec la règle suivante : 1 $ALUT \approx 1.5 \times LC$ et 1 $LC = 30$ *to* 40 *Gates* sur un FPGA de la famille Virtex. Les coûts associés au arcs i.e. NoC sont issus des Table 5 et 9.

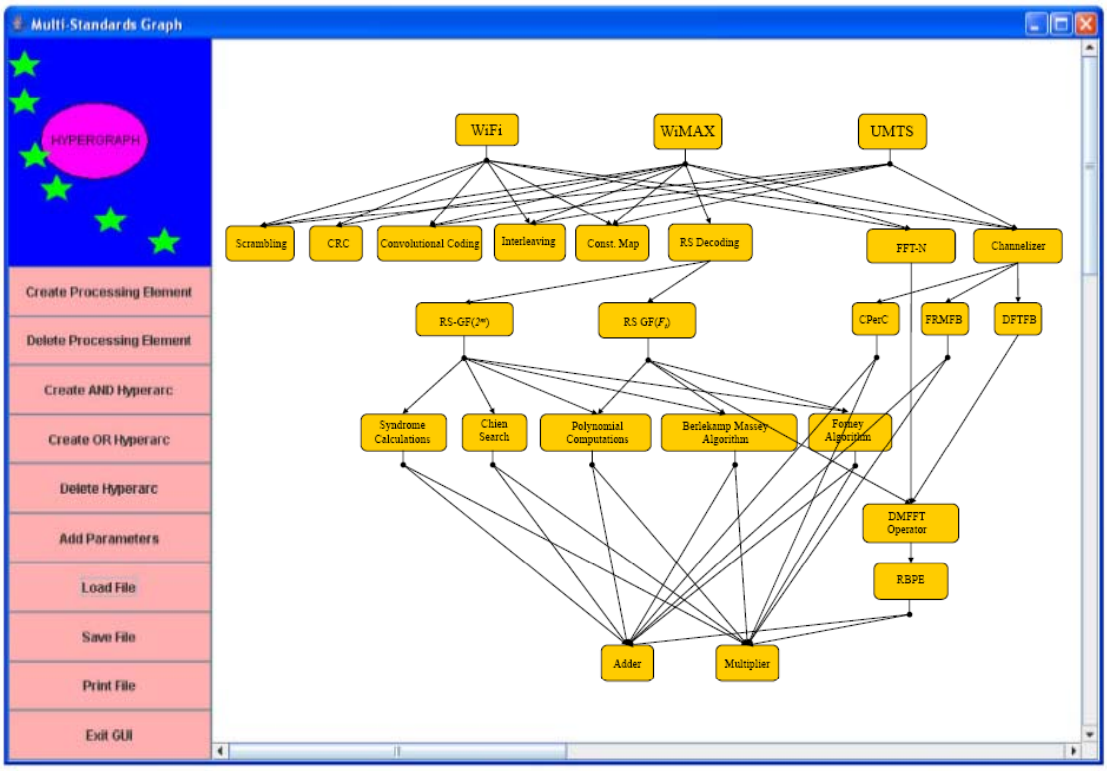

Figure 40: Copie d'écran du graphe dans l'interface graphique

Table 10: Cost parameters for different blocks of Fig. 41

Name of Function	**BC (ALUTs)**	**CC (ns)**
DMFFT	4857	5.5
RBPE	514	5.18
Re-Adder	26	2.37
Re-Multiplier	104	5.18
DFTFB	11868	77
FRMFB	7938	38.67

9.4 Résultats de l'optimisation

Après avoir lancée l'optimisation, les opérateurs RBPE, PloyCompt (*polynomial computation*), Berkelamp et Forney sont sélectionnés pour une implantation à coût minimal. La Fig. 42 résume à différentes étapes clef les résultats du processus d'optimisation.

Détaillons ces résultats. Le coût de l'équipement (reduit aux tris éléments de décodage RS, démodulation OFDM et canalisation) de la Fig.41 peut s'écrire :

$$\mathbf{C_{System}} = Cost_{RS_{Decoding}} + Cost_{OFDM} + Cost_{Channelizer} \quad (53)$$

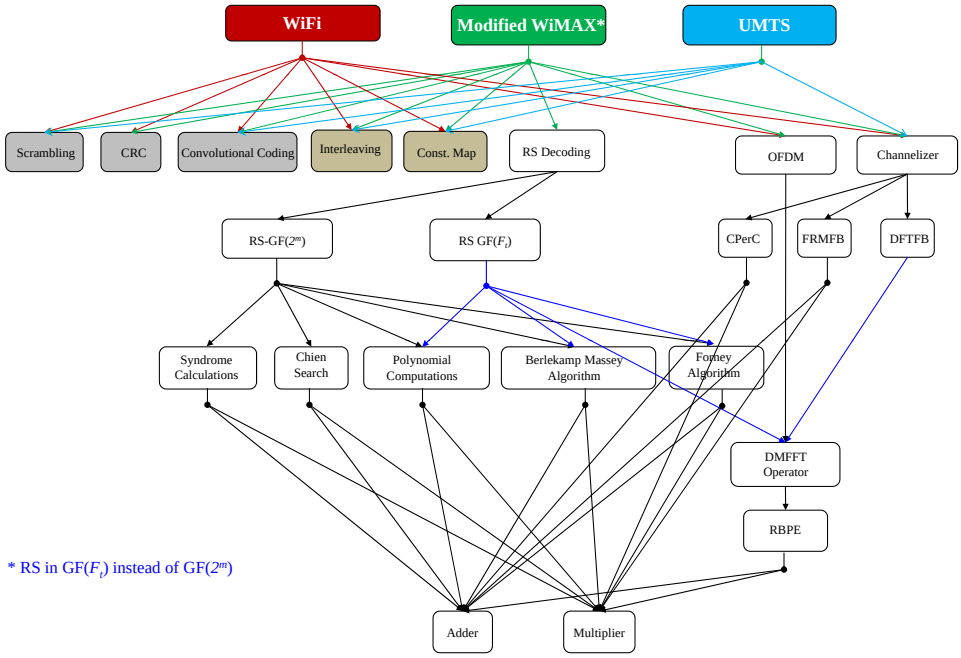

Figure 41: Version en gros plan de la Fig. 40

Trouvons le coût de chaque fonction de l'équation (53). Le coût du décodage RS (comme déjà mentionné) peut être tiré de l'équation (7) rappelée ici :

$$Cost_{level} = \Big(\sum_{i=1}^{i=m} CC_i NoC_i \Big) \times NoC_{i+1} + \sum_{i=1}^{i=m} BC_i.N_i \tag{54}$$

avec $N_i = 1$ pour les noeuds en blanc et $N_i = 0$ pour les noeuds en gris (non considérés) de la Fig.41. Si l'on considère le noeud *RS decoding*, alors $NoC_{i+1} = 1$ et l'équation (54) devient :

$$Cost_{RSDecoding} = \Big(\sum_{i=1}^{i=5} CC_i NoC_i \Big) \times NoC_{i+1} + \sum_{i=1}^{i=5} BC_i \tag{55}$$

En utilisant la première possibilité pour effectuer le décodage RS, l'équation (55) devient :

$$\begin{aligned} Cost_{RS-GF(2^m)} &= Cost_{Syndrome} + Cost_{Chien} + Cost_{Poly} \\ &+ Cost_{Berlekamp} + Cost_{Forney} \end{aligned} \tag{56}$$

soit en reprenant l'équation (27) du chapitre 6,

$$Cost_{RS-GF(2^m)} = \big(7064 \times 5.18 + 7048 \times 2.37\big) \times 1 + 104 + 26 = 53425.28 \tag{57}$$

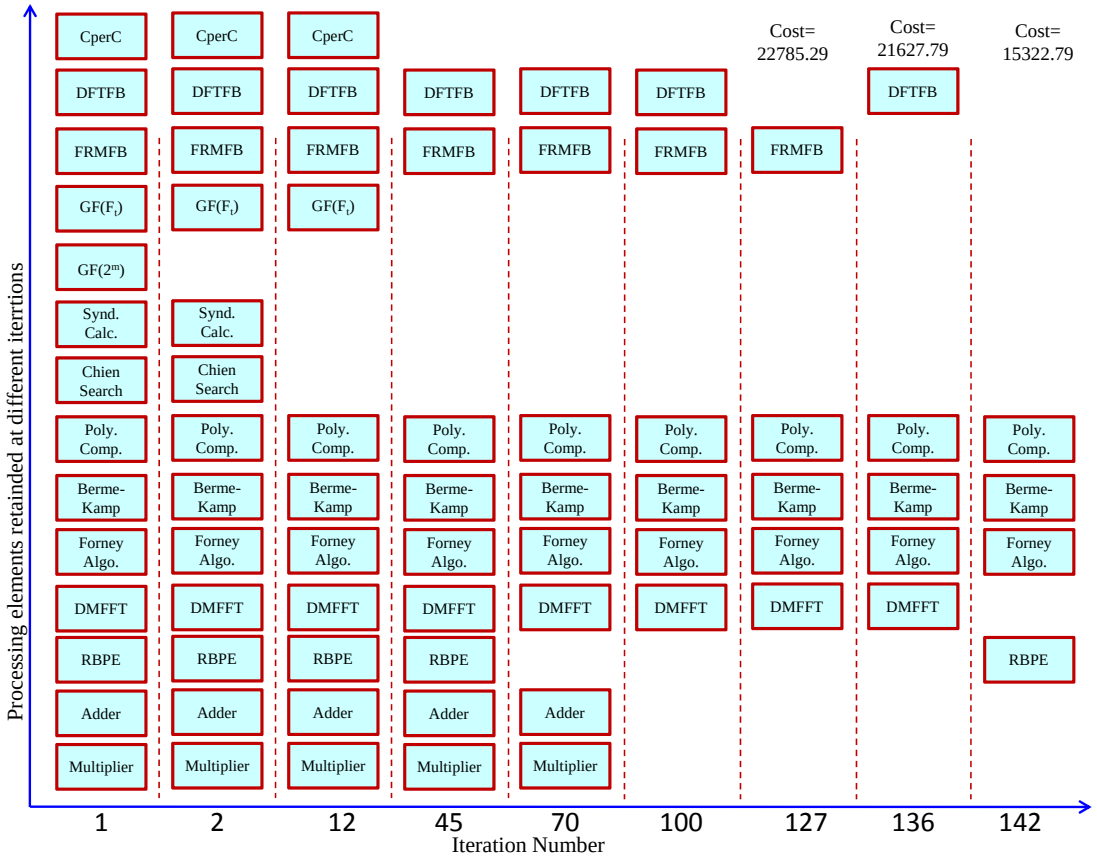

Figure 42: Number of operators kept at different number of iterations

Avec la seconde option pour le décodage RS, l'équation (55) devient :

$$Cost_{RS-GF(F_t)} = Cost_{Poly} + Cost_{Berlekamp} + Cost_{Forney} + Cost_{DMFFT} \tag{58}$$

soit en reprenant l'équation (30) du chapitre 6,

$$Cost_{RS-GF(F_t)} = \left(920 \times 5.18 + 904 \times 2.37 + 2 \times 5.5\right) \times 1 + 104 + 26 + 6819 = 13868.08 \tag{59}$$

Dans le cas du décodage RS, c'est donc la deuxième option, i.e. l'implantation dans $GF(F_t)$ qui minimise le coût.

Nous avons considéré jusqu'à présent la mise en oeuvre de RS dans $GF(F_t)$ via l'opérateur DMFFT. MAis si l'on pousse encore le raisonnement, la DMFFT peut elle-même être réalisée via l'opérateur RBPE et alors sont coût devient en utilisant l'équation (7) :

$$\begin{aligned} Cost_{DMFFT_{RBPE}} &= CC_{RBPE} \times NoC + BC_{RBPE} \\ &= 5.18 \times 128 + 514 \\ &= 1177.04 \end{aligned} \tag{60}$$

Le coût de la DMFFT implantée à l'aide du RBPE présente donc un coût minimal. En l'appliquant aussi au décodage RS, et en reprenant l'équation (58) pour laquelle on obtenait un coût de 13868,08, le coût du décodage RS devient :

$$Cost_{RS-GF(F_t)} = \big(920\times5.18+904\times2.37+2\times(128\times5.18)\big)+104+26+514 = 8878.16 \quad (61)$$

Considérons maintenant la démodulation OFDM. Elle requiert le noeud FFT-N qui est réalisé par l'opérateur DMFFT. Donc le coût de la démodulation OFDM est de :

$$\begin{aligned} Cost_{OFDM} &= Cost_{DMFFT} = BC_{DMFFT} + CC_{DMFFT} && (62)\\ &= 6819 + 5.5 = 6824.5 \\ & && (63)\end{aligned}$$

Mais la DMFFT est déjà réalisée avec le RBPE pour le décoadeg RS. Il n'est donc pas nécessaire de repayer son coût d'achat BC et seul le coût d'exécution CC est à mettre à jour. Donc pour la démodulation OFDM $BC = 0$ et CC dépend de $NoC = 128$. Le coût de l'OFDM devient :

$$\begin{aligned} Cost_{OFDM} &= Cost_{DMFFT} = BC_{DMFFT} + CC_{DMFFT} && (64)\\ &= BC_{DMFFT_{RBPE}} + CC_{DMFFT_{RBPE}} \\ &= 0 + 1 \times (128 \times 5.18) \\ &= 663.04 && (65)\end{aligned}$$

La réduction du coût est manifeste entre l'équation (62) et l'équation (64) grâce à l'utilisation conjointe du RBPE pour le décodage RS et la démodulation OFDM.

La canalisation peut être effectué des trois manières exposées dans la in Fig. 41. Pour les options FRMFB et DFTFB les coûts du canaliseur sont :

$$\begin{aligned} Cost_{Channelizer_{DFTFB}} &= BC_{DFTFB} + CC_{DFTFB} && (66)\\ &= 11868 + 77 = 11945 \end{aligned}$$

et,

$$\begin{aligned} Cost_{Channelizer_{FRMFB}} &= BC_{FRMFB} + CC_{FRMFB} && (67)\\ &= 7938 + 38.67 = 7976.67 \end{aligned}$$

On retrouve donc bien le résultat du chapitre 8, à savoir que lorsque la canalisation est considérée seule, le coût du FRMFB est inférieur à celui du DFTFB.

Maintenant, nous considérons la conception entière. Dans ce cas, le DFTFB peut utiliser l'opérateur DMFFT qui est par ailleurs indispensable pour la démodulation OFDM. Le décodage RS dans $GF(F_t)$ donnant un coût inférieur pour cette partie décodage, cela

renforce encore l'opérateur DMFFT, via le RBPE. En présence de la DMFFT le coût d'achat BC du DFTFB est réduit à $11868 - Cost_{DMFFT} = 5043.5$ ce qui le place à un coût inférieur à celui de la FRMFB (which is 7976.67), et alors le coût du canaliseur DFTFB utilisant l'opérateur RBPE est :

$$\begin{aligned} Cost_{Channelizer_{DFTFB}} &= BC_{DFTFB} + CC_{DFTFB} \\ &= BC_{DFTFB} + 77 + CC_{DMFFT} \\ &= 5043.5 + 77 + 1 \times (128 \times 5.18) \\ &= 5783.54 \end{aligned} \tag{68}$$

(69)

En utilisant alors les équations (61), (64) et (68), l'équation (53) donne :

$$\begin{aligned} \mathbf{C_{System}} &= Cost_{RS-GF(F_t)} + Cost_{OFDM} + Cost_{Channelizer_{DFTFB}} \\ &+ 8876.16 + 663.09 + 5783.54 \\ &= 15322.79 \end{aligned} \tag{70}$$

C'est le résultat du coût de la conception dans une approche par opérateur commun résolue par notre méthode. Si l'on ne tient pas compte de cela, on retombe alors sur une approche de conception plus classique par optimisation locale et alors le coût de conception aurait été de :

$$\begin{aligned} \mathbf{C_{System}} &= Cost_{RS-GF(F_t)} + Cost_{OFDM} + Cost_{Channelizer} \\ &+ 13686.08 + 6824.5 + 7976.5 \\ &= 28669.25 \end{aligned} \tag{71}$$

La différence entre les deux coûts des équations (70) et (71) donne le gain obtenu grâce à l'approche de conception par opérateurs communs.

9.5 Resultats et conclusion

Ce chapitre résume la philosophie de la solution d'exploration architecturale proposée dans cette thèse, à savoir fournir les moyen à un concepteur de trouver un optimal global en termes d'opérateurs communs pour un équipement SDR multi-standards. Dans le chapitre 6, nous avons relevé l'intérêt de l'opérateur DMFFT pour la modulation OFDM et le codage canal, et dans le chapitre 8 celui du FRMFB pour la canalisation. Dans un graphe plus complet, nous montrons dans ce chapitre que cela peut changer les équilibres et remettre en cause les optimums trouvés localement. C'est tout l'intérêt de ce travail de thèse. Nous n'avons pas étudié de graphe plus complexe encore en raison du manque de chiffres d'implantation sur cibles. Il n'était pas non plus possible d'envisager développer des chaînes entières de communication (en VHDL par exemple), ce qui était en-dehors des objectifs de cette thèse qui devait rester concentrée sur la méthode de conception.

Conclusion générale et perspectives

Ce manuscrit a présenté mon travail de recherche effectué au sein de l'équipe SCEE de SUPELEC/IETR sur l'optimisation d'équipements SDR multi-standards en utilisant une approche théorique. Cette approche, qui est basée sur un formalisme de graphe, fait partie de la catégories des techniques de paramétrisation par opérateurs communs.

Nous avons commencé le manuscrit en situant notre problème de conception au sein du domaine de la radio logicielle dans le chapitre 1. L'approche de conception de type *velcro* encore majoritairement utilisée ne sera bientôt plus viable. D'où la nécessité d'introduire de nouvelles méthodes d'optimisation de la conception, en particulier d'équipements multi-standards, qui vont se généraliser.

Partant de ce constat, le chapitre 2 détaille l'approche de conception par paramétrisation et met en particulier en valeur l'approches dite par opérateurs communs. Cela consiste à étudier les opérations du système à plusieurs niveaux de granularité. La sélection du niveau optimal permet de faire l'équilibre entre complexité et performance. Une représentation du problème sous la forme d'un graphe est retenue.

Dans le chapitre 3, les caractéristiques du graphe permettant de modéliser les traitements effectué par un équipement de radio communication sont définies.

Le chapitre 4 étudie les différentes possibilités en termes de paramètres de coût qu'il faut ajouter pour annoter le graphe, en fonction des composants cibles visés. La fonction de coût associée au graphe est aussi définie. Une interface graphique a été développée pour permettre de construire les graphes de l'étude.

Une étude de différentes techniques d'optimisation fait l'objet du chapitre 5 et les confronte à notre problème d'optimisation. Le choix d'un algorithme de recuit simulé y est expliqué.

Les chapitres 6, 7, 8 et 9 des exemples de problèmes de conception sont traités pour des problèmes restreints à un sous-graphe. Cela permet de valider l'approche pour le cas des opérateurs DMFFT et LFSR notamment. Enfin, un dernier exemple tend à montrer le problème de manière plus complète et comment l'approche de conception proposée s'applique à un cas de conception plus étendu, ce qui est sa raison d'être.

Les perspectives à la suite de ce travail sont très nombreuses. Tous les problèmes n'ont d'une part pas été résolus, et d'autre part de nombreux axes de recherches ont été identifiés pour de futures investigations. En voici les plus importants :

- La solution proposée vise à ré-utiliser des opérateurs communs. Par conséquent un problème d'ordonnancement en découle automatiquement. Ceci a été volontairement éludé dans nos travaux en raison de nombreux autres points à régler en priorité. Cependant, nous pouvons déjà y apporter cet éclairage. A un premier niveau, tant que les opérateurs peuvent répondre en temps-réel quel que soit le nombre d'opérations faisant appel à eux, le problème se ramène à un cas conventionnel. Soit

des solutions d'ordonnancement statique sont envisagées (par période temporelle correspondant à la durée de vie d'un standard), soit des solutions d'ordonnancement en temps-réel exécutée par un processus gestionnaire de l'équipement (par exemple un processeur ayant un RTOS). A un second niveau, si un opérateur est surchargé, une solution consiste à dupliquer l'opérateur, une autre à changer le jeu d'opérateurs sélectionné. Il faudrait que cela soit pris en compte dans la méthodologie, ce qui semble nécessiter au moins un autre travail de thèse.

- De nouveaux algorithmes d'optimisation seraient à investiguer. D'autant plus si le problème étudié ici est combiné avec celui de l'ordonnancement évoqué ci-dessus. Notamment cela replacerait les algorithmes génétiques en position de prétendant.

- Un travail très important pour continuer ce travail est d'identifier de nouveaux opérateurs communs qui soient existent déjà soit sont à inventer.

- Des graphes de plus en plus complets, c'est-à-dire incluant plus de fonctions d'un même standard et plus de standard, sont à construire. Nous n'avons regardé que des sous-graphes de la partie couche physique jusqu'à présent. L'un des buts de cette approche est justement de pouvoir trouver des opérateurs communs à des traitements des différentes couches OSI afin d'effectuer de l'optimisation inter-couches au niveau des traitements.

- Il pourrait être aussi possible de reformuler la fonction de coût. Nous avons utilisé une somme pondérée mais d'autre possibilités existent.

- Un problème en ce qui concerne les coûts a été identifié en particulier dans le cas d'une conception hétérogène impliquant du matériel (de type FPGA ou ASIC par exemple) et du logiciel (de type processeur ou DSP par exemple). Cela requiert une analyse complémentaire. Résoudre ce problème serait très prometteur car la méthode proposée permettrait alors de'effectuer un partitionnement automatique matériel/logiciel.

- Une perspective à long terme est de fournir un outil industriel aux concepteurs d'équipements SDR multi-standards qui leur permette d'optimiser leur conception et d'orienter leur choix en termes de granularité en fonction des intérêts de leur entreprise. En effet, suivant qu'ils font partie d'une société de micro-électronique ou celle d'un fabricant de processeur, le choix en terms de granularité peut varier considérablement.

Chacun de ces sujets mériterait une thèse au moins pour y répondre ou commencer à y répondre.

General Introduction

1 Background and context

Wireless signal processing technology is experiencing a period of radical change, for both handsets and basestation applications. There has been an enormous proliferation of standards in broadcast television, radio and mobile communications in a short duration of time. At present we have two or three standards at maximum, in our mobile phones like Global System for Mobile Communication (GSM), Universal Mobile Telecommunication System (UMTS) and Bluetooth etc. In the near future we will have in our mobile phones, standards like Wireless Fidelity (WiFi), Worldwide Interoperability for Microwave Access (WiMAX) for Wireless Local Area Networks (WLAN), Galileo and Global Positioning System (GPS) for positioning, Digital Video Broadcasting-Handheld/Terrestrial (DVB-H/T) for digital video broadcast, Digital Radio Mondiale (DRM) for digital radio etc. to name a few. Further, the rate of technology innovation is also accelerating and predicting technological change. These standards form the basis for an ever-growing number of sophisticated consumer electronic devices, each with the potential to sell in very high volumes. A multi-standards system is the right solution to successfully communicate with different systems using different air-interfaces. Multi-standard systems can work within one standard family e.g. UMTS Terrestrial Radio Access-Frequency Division Duplexing/Time Division Duplexing (UTRA-FDD/TDD) for UMTS, or across different networks e.g. Digital Enhanced Cordless Telecommunications (DECT), GSM, UMTS, WLAN etc. The need for transparency, i.e., the ability of radios to operate with some, preferably all, of these standards in different geographical regions of the world is ever increasing.

In conventional multi-standards designs, these complex standards are implemented using dedicated architectures, which are optimized to reduce cost to the absolute minimum. Products developed using dedicated architectures are often difficult to upgrade in order to support changes in the standards or to add new features.

In the beginning of 90's, the concept of *Software Radio* (SR) emerged from demonstrations in military research to become a cornerstone of the 3G strategy for affordable, ubiquitous and global communications. In addition, the need for a *multi-band* system which supports more than one frequency band used by a wireless standard (e.g., GSM 900, GSM 1800, GSM 1900), and a *multi-channel* system that supports two or more independent transmission and reception channels at the same time has cherished the growth of the SR concept.

The SR technology is a way to design a sufficiently programmable and reconfigurable architecture able to support many different transmission standards on a common platform. The reconfigurability of a SR system can offer a range of benefits at different levels. A radio system implemented on a reconfigurable architecture can be upgraded to fix bugs or to add functionalities, and it can also support new standards as it is assumed that there is sufficient flexibility in the architecture. Demand for a *multi-services* system, which provides different services e.g. telephony, data, video streaming, GPS and soon handheld television (TV) broadcast, Information Technology Services (ITS) etc. is rising. With the development of short-range networks like Bluetooth and IEEE 802.11, it is now possible to enhance the services of a radio by leveraging other devices that provide complementary services. Software radio's reconfiguration capability will enable to support an almost infinite variety of service capabilities in a system.

The communication chains of different air interface standards, intended to be implemented on a common platform, have some common signal processing operations such as channel coding, modulation, equalization, etc. In order to exploit to a great advantage the commonalities among these communication tasks for different standards, one need firstly to identify these commonalities and secondly find the optimal way to implement a generic hardware platform with programmable modules. In this sense, a technique called *parametrisation* was introduced. Parametrisation is a very promising technique that consists of designing radio systems entities in a way which permits to take advantage of the programmable or at least reconfigurable capabilities of the underlying hardware of SR systems. The key idea is to get an optimal sharing between hardware and software resources and find a best way to reuse some hardware and software modules that can be adapted just by a parameter change without affecting the system's performances.

This thesis aims to contribute to the design of multi-standard SR equipment from a particular angle compared to research efforts in general, carried out in SR community. Indeed, it does not propose to study how to adapt the current design tools for new challenges proposed by the *software defined radio* (hardware/software co-design, high-level abstraction tools, etc.) but how to re-organize the processing chain, in order to exploit new degrees of freedom in design. It fits into the context of SR and more precisely in the Common Operators' (CO) technique for parametrisation. Further it explores in detail, two approaches of CO's technique; *The Pragmatic Approach* in general and *The Theoretical Approach* in particular. CO technique can help reduce the cost, size and power consumption of SDR equipments, which are required to support a wide range of existing and future wireless technologies and services.

2 Thesis outline

The manuscript of this thesis is divided into three parts.

Part-I consists of two chapters.

Chapter 1 provides an introduction to SR technology in general and outlines some of the limitations associated with it. This chapter further describes what software defined radio

(SDR) is, plus various commercial and military efforts put in this regard. It also looks into various transceiver architectures. It highlights the need/drivers for SR and different challenges in designing SR. In addition to this, it presents our point of view of tackling the design of multi-standards SDR equipment as a problem of granularity adjustment. The SR's problematic, in this thesis is considered from the viewpoint of a research of commonalities across several standards and in the standards themselves. In this context, an important technique called *parametrisation* that aims to optimize the resources use in the SR system is presented in detail in chapter 2.

Chapter 2 is about the *parametrisation* techniques in SRs and in particular about CO's technique. According to the methodology of parametrisation, the common aspects of the different standards will become one common processing procedure, which could be installed in the device. This common procedure could be executed by a simple *call*, thereby resulting in a gain of both size and time with respect to the amount of code to be downloaded or read (only parameters) in order to modify the radio behavior. From the design point of view, the parametrisation will optimize the co-design and will probably reduce the time to market by enabling incremental design. The parametrisation techniques can be divided into two categories 1) Common Function Technique and 2) CO's Technique. CO's technique can be further divided into: 1) *The Pragmatic Approach* and 2) *The Theoretical Approach*. It is explained how the *theoretical* approach to find the common operators in the graph model of multi-standards SDR equipment can help the SDR designer, in finding global optimality. Various criteria for measuring this optimality could be memory size, downloading speed, reconfiguration speed, computation complexity etc. Chapter 1, 2 form the foundation for the development of solution to multi-standards design problem presented in Part-II of thesis.

Part-II consist of three chapters and these chapters explain different steps to solve the multi-standards SDR design problem using theoretical approach in the context of CO's technique for parametrisation described in Part-I of this manuscript. These steps include 1) The definition of the graph 2) The definition of cost parameters and development of the cost function and 3) The optimization algorithms.

Chapter 3 explains how to model air interface standards as graphs. Starting from the fundamentals of graph theory and after exploring the already present graphs it is concluded that for the multi-standards SDR design problem, the best choice is to use hypergraph. This is because of the dependencies needed for problem which were not available for other graph types. After this it describes the development of graph theoretic models for multi-standards SDR systems. Simplified versions of graphs for tri-standards SDR equipments are presented. In addition, because of availability of extensive literature in the domain of network theory this chapter illustrate a way to recast the problem as network design problem. Later because of some difficulties it is decided not to convert the graph into a network design problem.

Chapter 4 looks for different cost parameters and types of the costs. Various costs parameters are identified and explained. Based on the selected costs associated with different graph entities, a cost/objective function is formulated that is to be solved by optimiza-

tion techniques which are presented in chapter 5. The objective function is formulated as multi-objective optimization problem (MOP) also known as multi-performance/vector optimization problem. Many real-world scientific and engineering MOPs are irregular and hence deterministic search techniques are not suitable for them. To solve these irregular problems, stochastic search and optimization approaches such as Simulated Annealing, Monte Carlo methods, Tabu search and Evolutionary Computation Algorithms were developed as alternative approaches. A graphical user interface is being developed in Java, to facilitate drawing of graphs and annotating the cost parameters and costs to various entities of graphs.

Chapter 5 discusses the selected optimization techniques. In chapter 5 of this part, various optimization techniques available in literature with their pros and corns are explained. Based on this investigation, some techniques are selected that seem to be most appropriate to the problem at hand. A tool based on these techniques is developed in C++, that can be used to solve the problem. Also presented is a generic design example with the costs based on intuition. This example is solved by the selected techniques of optimizations implemented by tool. The objective is to find the best suited technique. Results proved that *simulated annealing* technique is better than other selected techniques.

Part III consists of four chapters and these chapters are composed of design scenarios based on different common operators.

Chapter 6 describes the use of dual mode fast Fourier transform (FFT) operator as CO. This chapter investigates the frequency processing of Reed Solomon (RS) codes with the aim to insert the classical FFT operator, initially used for complex Fourier transform, in the encoding and decoding processes of RS codes. To be able to exploit the whole FFT-$\mathbb{C}$ structure, it explores finite field transforms having a highly composite length. The candidate transforms are the Fermat transforms defined over $GF(F_t)$. These transforms led us to revive a specific class of RS codes defined over $GF(F_t)$. The redesign of the FFT-$\mathbb{C}$ is elaborated in such a way to be able to provide two functionalities: complex Fourier transform and Fermat number transform (FNT). Conceptually, the design of such a dual mode operator implies the design of arithmetical operators capable to operate over the two domains: $\mathbb{C}$ and $GF(F_t)$. Proposed in this chapter, is a reconfigurable architecture for the butterfly that constitutes the core of the dual mode operator. Based on the FFT-$\mathbb{C}$ structural strategy, the architecture of the dual mode FFT (DMFFT) operator is designed. To evaluate the complexity and speed performances of this operator, DMFFT is implemented on ALTERA's Field Programmable Gate Array (FPGA) devices. We illustrate our design approach through a simplified sub-design example. Results show that RBPE offers minimal cost when designing of multi-standard systems is considered. Compared to a Velcro FFT/FNT operator, the DMFFT presented an important gain in terms of Adaptive Lookup Table (ALUTs) and memory saving.

Chapter 7 explains the use of linear feedback shift register (LFSR) and its different variants as COs, that can be used to perform various tasks in an SDR equipment. Use of LFSR as a CO is one of the first implementation of CO's technique on the area of techniques of parameterizations. Reconfigurable LFSR (R-LFSR) and Extended Reconfigurable LFSR

(ER-LFSR) which are derived from an improved classical LFSR structure, can be configured to implement functions such as *pseudo random sequence generation*, *scrambling*, *convolutional coding*, *cyclic coding* and *RS coding/decoding*. These developed LFSRs replace functions that can be derived from LFSR operations and are particularly suited for Orthogonal Frequency Division Multiplexing (OFDM) based air interface standards like IEEE 802.11, IEEE 802.16 and 3GPP LTE, etc. We explain our four developed common architectures to design 16 LFSR COs that can carry out several operations of multi-standard SDR transceivers. We also give the implementation details on Altera Cyclone II. Either a single or combination of these LFSR operators can be used to perform the functions as necessitated by the different air interfaces to be supported by SDR. Further we illustrate our design approach through a simplified sub-design and by running the optimization algorithms on subpart of the graph. We prove that different common operators are chosen depending upon the considered scenarios and they give efficient solutions as compared to *Velcro* methods.

Chapter 8 presents and compares various channelization techniques in order to find which one is best suited to SDR applications. In the context of multi-standard SDR equipments, channelization is an extremely important task, which can be performed by a variety of filter banks. Channelizer is the most computationally intensive part of an SDR receiver since it operates at the highest sampling rate. It extracts multiple narrow-band channels from a wide-band signal. Low complexity, high-speed and reconfigurable channelizers are required in the SDR receivers. The flexibility of an SDR depends on its capability to operate in multi-standard wireless communication environments. To evaluate the complexity of various filter bank techniques, we perform a quantitative analysis based on the underlying mathematical equations/formulas in the presented design scenarios. Compared to a *Velcro* approach (PC approach) and the discrete Fourier transform filter bank (DFTFB) approach, the frequency response masking filter bank (FRMFB) approach presents an important gain in terms of number of multiplier and adders.

Chapter 9 presents a case study of a complex graph. It presents the idea of looking into the design of multi-standards equipment from a system level point of view. It integrates more than one CO in the graph model and then seeks for the opportunities of finding the global optimum. In the previous three chapters the focus was on the sub-parts of a global graph. This chapter extends the research towards the ultimate aim of developing an optimized and complete graph.
A section of general conclusions is provided at the end, which summarizes the major results obtained in this thesis, and outlines possible future work in this field.

3 Contributions

This thesis makes several distinct contributions to the literature:

1. We examine the impact of parametrisation techniques on design of multi-standards SDR equipment. Our focus is on the common operator's technique of parametrisation. We particularly target on the theoretical approach under the CO technique. The results were presented in:

S. T. Gul, L. Alaus, D. Noguet, C. Moy and J. Palicot "The Common Operator Technique: An Optimization Process to Identify and Design a Set of Common Operators to Perform SDR Equipment (project NEWCOM++)," Dynamic Spectrum Management in Cognitive Radio Networks, ICT-MobileSummit 2009, Santander, Spain, June 2009.

2. We explain the modeling of multi-standard SDR systems as hypergraphs. A GUI has been developed in Java to help in drawing graph models. Graph modeling of multi-standard SDR systems was presented in:

   S. T. Gul, C. Moy and J. Palicot, "Two scenarios of flexible multi-standard architecture designs using a multi-granularity exploration," The 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communication-PIMRC'07, Athens, Greece, pp. 1-5, September 2007.

3. We illustrate the conversion of simple hypergraphs to weighted hypergraphs. In this context we discuss various possibilities for cost parameters and costs that can be associated with entities of graph. By doing this we convert our the graph models to an optimization problem. Results were presented in:

   S. T. Gul, C. Moy and J. Palicot, "Graphical Modeling and Optimization of Air Interface Standards for Software Defined Radios," 12th IEEE International Multitopic Conference-INMIC2008, Karachi, Pakistan, pp. 473-479, December 2008.

4. We investigate various optimization techniques that can be applied to solve the graph optimization problem. We demonstrate the effectiveness of selected optimization techniques with examples. By using the optimization tool developed in C++ we perform a comparison to check which of the selected techniques is most suitable for our problem.

5. We illustrate the use of DMFFT operator as a common operator. We present the use of this DMFFT operator in OFDM demodulation and RS decoding in a theoretical approach based on graphical formalism. Results are accepted to be published in:

   S. T. Gul, Ali Al-Ghouwayel, C. Moy and Y. Louët, "A Novel Design of Reconfigurable Fourier Transform Operator Over C and GF(Ft) for Future Multi-standards SDR Equipments," International Journal of Communication Networks and Distributed Systems-IJCNDS, Accepted in June 2009 for publication by the end of 2009.

6. We explain the use of LFSR operator as a common operator. We elaborate how we can use a single LFSR operator or a combination of different LFSRs, depending upon the standards to be supported and functionalities needed to be preformed by LFSRs. Results of this work are accepted to be published in:

   S. T. Gul, L. Alaus, C. Moy, J. Palicot and D. Noguet, "Optimal set of LFSR Common Operators for Multi-Standards Cognitive Radio Terminals," International Journal of Autonomous and Adaptive Communications-IJAACS, Special Issue on Cognitive Radio Systems, Accepted in July 2009 for publication in 2010.

7. We examine the various channelization techniques and use of FRMFB as a channelizer that can be as a common operator in the SDR design. Results were published in:

S. T. Gul, R. Mahesh, C. Moy, A. P. Vinod and J. Palicot, "A Graphical Approach for the Optimization of SDR Channelizers," International Union of Radio Science (Union Radio Scientifique Internationale-URSI); XXIX General Assembly, Chicago, Illinois, USA, August 2008.

and are also submitted in:

S. T. Gul, R. Mahesh, C. Moy, J. Palicot and A. P. Vinod "Filter Bank Techniques for SDR Channelizers and their Optimization using Graph Theoretical Approach," EURASIP Journal on Advances in Signal Processing, Special Issue on Filter Banks for Next Generation Multicarrier Wireless Communications, Submitted, June, 2009.

Part I

Introduction to Part-I

Increasing demands of users for flexibility, scalability, multi-functional and multi-service communication equipments has motivated the need towards the potential realization of software radio (SR) technology. We can envision that SR as a technology will help bring together various forms of communications. The incorporation of mobile communications, broadcast receivers, multimedia, personal computing, internet is possible with the help of SRs.

Let us start with the first part of this thesis that consists of two chapters. Chapter 1 highlights the emergence of SR technology. This chapter starts with an introduction to SR technology and its practical version i.e. software defined radio (SDR). We discuss different transceiver architectures with their advantages and limitations. We explain different challenges and drivers for SDR. We present our point of view of considering the design of multi-standards SDR equipment as a granularity adjustment problem. Then we present SDR projects' survey that shows, how the SR concept is developing and identifies key concepts and technologies useful for the practical implementation of a software radio. This chapter ends with a conclusions' section.

Chapter 2 discusses the parametrisation techniques for SDR. In SDR, parametrisation is a very promising technique that consists of designing radio systems entities in a way which permits to take advantage of the programmable or at least reconfigurable capabilities of the underlying hardware of SDR systems. We begin this chapter with the fundamentals of techniques of parametrisation. In this context we explain with examples, two important techniques of parametrisation. Then we describe two approaches under the common operators' (CO) technique of parametrisation namely, pragmatic approach and theoretical approach. The later approach forms the core of this thesis. Finally a conclusions' section ends this chapter.

Chapter 1

Emergence of Software Radio

Contents

This chapter presents the development of software radio (SR) technology from functional and historical perspectives. It then examines the evolving spectrum of requirements from mobile devices that resulted in the emergence of Software Defined Radio (SDR). The requirement challenges are then discussed in the context of currently available (and emerging) technologies for implementing SDR processing subsystems.

1.1 Introduction

With the emergence of new standards and protocols, wireless communications is developing at a furious pace. The challenge in creating sophisticated wireless Internet connectivity is compounded by the desire for future-proof radios, which keep radio hardware and software from becoming obsolete as new standards, techniques, and technology become available. The concept of integrated seamless global coverage requires that the radio supports two distinct features [2]:

1. Global roaming or seamless coverage across geographical regions
2. Interfacing with different systems and standards to provide seamless services at a fixed location

Multi-mode phones that can switch between different cellular standards like GSM and UMTS fall in the first category, while the ability to interface with other services like Bluetooth or IEEE 802.11 networks falls in the second category. Further, the rate of technology innovation is accelerating and predicting technological change. As a result, to keep their systems up to date, wireless systems manufacturers and service providers must respond to changes as they occur by upgrading systems to incorporate the latest innovations or to fix bugs as they are discovered. Since frequent redesign is expensive, time-consuming, and inconvenient to end users, interest is increasing in future-proof radios. The desire to reduce the cost, size and power consumption, and to make devices easier to manage in the field while taking full advantage of technology improvements wherever possible, has driven the technology evolution path we are still on today.

Existing technologies for voice, video, and data use different packet structures, data types, and signal processing techniques. Integrated services can be obtained with either a single device capable of delivering various services or with a radio that can communicate with devices providing complementary services. The supporting technologies and networks that the radio might have to use can vary with the physical location of the user. To successfully communicate with different systems, the radio has to communicate and decode the signals of devices using different air-interfaces.

Reconfigurability became familiar to many radio researchers with the publication of special issue on software radios [53]. Reconfigurability denotes the potential capability of a system to be changed. First step consists in changing at design time. Second step is during operation. First step is a natural evolution in a radio design while second step is a new feature brought by software radio. However, in the context of wireless communication reconfigurability tackles the changeable behavior of wireless networks and associated equipment, specifically in the fields of radio spectrum, radio access technologies, protocol stacks, and application services. Such radios can be implemented efficiently using SR architectures in which the radio is reconfigured, based on the system it will be interfacing with and the functionalities it will be supporting [54].

Today, wireless communication research is in its Fourth Generation (4G) phase. Second-generation (2G) wireless technology consists of a handful of incompatible standards, and the goal behind the development of third-generation (3G) standards was compatibility

among these standards within and between different generations' standards. Even if cellular standards globally converge, 3G systems require multi-mode operation and automatic mode selection.
The Third Generation Wireless age (3G) has provided an increase in data rate to the users which allows them to experience more than just voice over the air. The International Telecommunications Union (ITU) proposes for next steps a maximum data rate of 100Mbps for high mobility situations and 1Gbps for stationary and low mobility situations like hot spots. These targets are being used by most research on 4G today.

With 4G the user's application will likely have the ability to control the quality of service and obtain a higher QoS for a higher cost. Higher QoS can be achieved through priority scheduling of packets, changes in data packaging, improved protection coding, better channel equalization techniques, implementation of smart antennas, and so on. The mobile subscriber must have the ability to select the network provider as well as the services needed. It is also envisioned that 4G will include earlier standards and their protocols, and that they will work harmoniously together. In both cases SR will be an irrefutable technology to achieve 4G goals and will contribute to the convergence of different solutions, each dedicated to a certain situation of mobility, coverage, data rate, QoS and cost. SR solutions can help reduce the cost, size of systems, which are required to support such a wide range of existing and future wireless technologies and services [55].

1.2 Software radio

The term software radio was coined by Joe Mitola in 1991 to refer to the class of reprogrammable or reconfigurable radios [56]. In other words, the same piece of hardware can perform different functions at different times.

A radio that has defined, in software, its modulation, error correction, and encryption processes, exhibits some control over the RF hardware, and can be reprogrammed is termed a SR. A good working definition of a SR is a radio that is substantially defined in software and whose physical layer behavior can be significantly altered through changes to its software [2]. The degree of reconfigurability is largely determined by a complex interaction between a number of common issues in radio design, including systems engineering, antenna, RF electronics, baseband processing, speed and reconfigurability of the hardware, and power supply management.

The SDR Forum defines different levels of SR as listed in Table 1.1. According to SDR Forum, the ultimate software radio (USR) is defined as a radio that accepts fully programmable traffic and control information and supports a broad range of frequencies, air-interfaces, and applications software. The user can switch from one air-interface format to another in milliseconds, use the Global Positioning System (GPS) for location, store money using smart card technology, or watch a local broadcast station or receive a satellite transmission. USR is defined for comparison purposes only.

In a nutshell, we refer to a transceiver as a software radio if its communication functions are realized as programs running on a suitable processor. Based on the same hardware,

Table 1.1: Levels of SDR

Tier	Name	Description
Tier 0	Hardware Radio (HR)	Implemented Using hardware components. Cannot be modified.
Tier 1	Software Controlled Radio (SCR)	Only control functions are implemented in software, e.g. inter-connects, power levels, etc.
Tier 2	Software Defined Radio (SDR)	Software control of variety of modulation techniques, wide-band or narrow-band operation, security functions, etc.
Tier 3	Ideal Software Radio (ISR)	Programmability extends to the entire systems with analog conversion only at the antenna.
Tier 4	Ultimate Software Radio (USR)	Defined for comparison purposes only.

different transmitter/receiver algorithms, which build radio transmission chains of different standards, are implemented in software. An ideal SR (ISR) directly samples the antenna output. A software defined radio (SDR) is a practical version of an SR: the received signals are sampled after a suitable band selection filter, thus incorporating some analog front end. A SR transceiver comprises all the layers of a communication system. The discussion in this thesis, however, mainly concerns the physical layer (PHY) but is compatible with the entire protocol stack of a radio communication system. A detailed discussion relating the performance of enabling hardware technologies to SR requirements, portending a decade of shift from hardware radios towards software intensive approaches can be found in [57, 58].

The concept of software radio, originally conceived for military applications has attracted much attention lately in the context of commercial applications [54, 59, 60].

The drivers of these two applications are somewhat different. The main driver for the military software radio is the need for a single radio which can communicate with the many types of military radios that use different RF bands and different modulation schemes [59]. In the commercial arena, software radio applications can be divided into user terminals and radio base stations. The computational load of software radios is of billion of operations per second [54] depending upon the radio standards; hence given today's state of the art implementation hardware technologies, it is extremely challenging to implement software radio at an acceptable cost and acceptable power consumption.

Moreover, in a multi-standards cellular environment, with the rapid growth in demand for cellular services and with the proliferation of micro-cells and pico-cells, there is tremendous interest in reducing the cost, complexity and size of radio terminals/base stations.

A great number of important contributions on software defined radio can be found in [53, 54, 61]). SDR is both a product of, and comes to the market in the context of dramatic changes in the worlds' information technology and usage environments [62].

Significant work on the reconfigurability issue of SDR can be found in the literature [63, 64, 65, 66, 67].

However, many authors narrow down their research interest to one particular aspect of the signal processing chain as SDR requires expertise in diverse domains. Explanations of sample rate adaptation techniques can be see in [68, 69, 70]

The design of the RF component of SDR is a challenging task. The wideband linear transmitter is one of the critical components in the flexible mobile terminals. This transmitter will be required to operate at frequencies extending from 900MHz to 5.5GHz covering the GSM, DECT, UTRA, Bluetooth and HIPERLAN/2 bands. High output power, linearity, efficiency and carrier frequency accuracy are some of the important parameters to be considered for this transmitter in all the frequency bands. As most of the transmitter designs previously reported are of a narrowband nature, an SDR transmitter will need to be a state-of-the art design. The transmitter proposed in [71] for TRUST project has three main components: a two-stage up-converter, a local oscillator and a power amplifier. The power amplifier block is the one that imposes the strongest design challenges. It should guarantee high power, efficiency and linearity in different bands. Linearity over wide bandwidths is an issue also with receiver. Details of TRUST approach for transmitter and receiver design can be found in [71]. Basic architectural solutions derived with in the TRUST project are presented in [72]. Moreover, RF front-end design techniques and issues are discussed in [73, 74, 75, 76, 77].

An SDR receiver uses an ADC to digitize the analog signal in the receiver as close to the antenna as practical. A/D conversion and state-of-art in ADC technology can be found in [78, 79].

A functional clustering for software radio based on a multi-standards analysis of baseband processing tasks was proposed in [80]. Instead of designing multiple architectures for multiple standards, SR targets single transceiver architecture for multiple standards. In an entire multi-standards baseband chain, the functions are very different, hence various functions can be categorized into 3 classes [80] namely; *coding* class, *data structuring* class and *modulation* class. Each class of functions is dedicated to a specific reconfigurable hardware in order to optimize the implementations.

The coding class includes functions like cyclic, convolutional and turbo coding. These functions are generally based on a linear shift register structure for 1-bit data. Software implementations are possible but hardware ones are more efficient and consume fewer resources. A software implementation accelerated through efficient dedicated co-processors corresponds to a suitable architecture for this class of processing functions.

Data handling class combines functions that manipulate data packets. These functions are exclusively dedicated to data transfers who demand a lot of memory. This class of algorithms is largely control-oriented. The high processing flexibility offered by processors combined with arrays of memory blocks can be efficient solution for functions of this class.

The Modulation Class corresponds to the baseband functions which take place before transposition on frequency carrier. In this class, the functions are computation-oriented and data are often oversampled. High throughput is needed. A combination of FPGA, DSP and hardware accelerators can be used for this class.

The classification of functions and their consequences on hardware cluster is depicted in Fig. 1.1. The general goal in each class of processing is to group similar processing to maximize the reuse of hardware. Grouping functions into classes helps when dealing with configuration management.

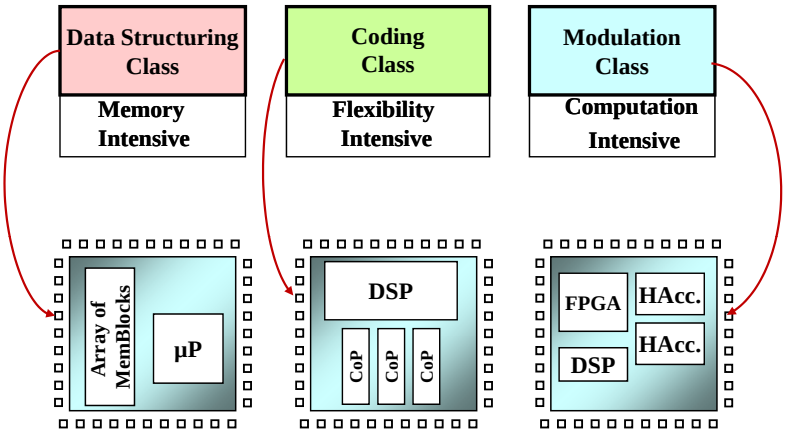

Figure 1.1: Classification of functions and their consequences on hardware cluster

A detailed discussion about channel coding techniques can be found in [81, 82, 83].

Notably, work related to signal processing in the digital baseband is centered around algorithms [84, 15]. One major contribution of Mitola [85] attempts to reexamine software radio from a truly unorthodox point of view, but his findings still pertain to algorithms and eventually do not reach beyond the Turing's theory of computing.

The motivation for introducing optimization aspects of flexible radio system design is the tremendous interest in reducing the cost, complexity and size of the radio terminal/base-station in the long term. The goal of this thesis is to focus on one particular point of the global SR issues and to establish general guidelines for optimizing multi-standards flexible SDR systems in order to give option to the SDR designer to orientate his/her design either toward Velcro or gate level primitive elements.

1.3 Transceiver architectures

At the highest level, all wireless equipments can be decomposed into four subsystems as shown in Fig. 1.2 [62]. The baseband subsystem sits between the RF front-end and the controller. It is responsible for end-user data encoding/decoding and signal modu-

lation/demodulation. The encoding/decoding function can be classified as a low speed signal-processing task and the modulation/demodulation task can be classified as a high speed signal-processing task.

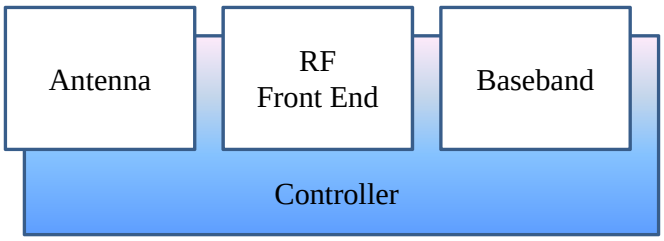

Figure 1.2: A top level view of wireless equipment architecture

The multi-band, multi-mode operation of a SR introduces stringent requirements on the underlying system architecture. The requirement of supporting multiple frequency bands affects the design of the Radio Frequency (RF) front end and the requirement of Analog-to-Digital (ADC) and Digital-to-Analog (DAC) converters [2]. The RF front end should be adjustable or directly suitable for different frequencies and bandwidths required by the different standards that the SR equipment intends to support. In the following subsections we discuss the various RF front end architectures and then we present the today's feasible SR architecture.

1.3.1 Radio frequency front end

Architecture of a typical digital radio system can be represented by the block diagram of Fig. 1.3. The transmitter is divided into an information source, a source encoder, an encryptor, a channel encoder, a modulator, a DAC and a RF front end block. Correspondingly, the receiver consists of an RF front end, an ADC, a synchronization block, a demodulator, a detector, a channel decoder, a decryptor and a source decoder. The main functions of RF front end are down and up conversion, channel selection, interference rejection and amplification. The exact point where the conversion between digital and analog waveforms is done depends on the architecture. In conventional radio architectures, the conversion is done at the baseband, whereas in some specific SR sub-architectures such as SDR that will be presented in subsection 1.3.6, the typical place for the ADC and DAC is between the stages of channel modulation, at an intermediate frequency. To be transformed into an ideal SR architecture, the architecture of the Fig. 1.3 must employ the digital processing block (i.e. the ADC and DAC) right beside the antenna which is currently feasible for very low RF systems, but rather impossible at a low cost and at a low power consumption for higher RF frequencies.

Although an ideal SR would have a very minimal Analog Front End (AFE), consisting of an ADC placed as close as possible to the antenna, any practical implementation still needs some analog parts of RF front end, and the design of a reconfigurable RF part remains a very complicated issue [2, 54]. The receiver section is more complex than the

transmitter and the ADC in particular is one of the most critical part limiting the choice of the RF front end architecture [2]. The transmitter side of the RF front end takes the signal from the DAC, converts the signal to the transmission radio frequency, amplifies the signal to a desired level, limits the bandwidth of the signal by filtering in order to avoid interference and feeds the signal to the antenna [14]. The receiver side converts the signal from the antenna to a lower center frequency such that the new frequency range is compatible with the ADC, filters out noise and undesired channels and amplifies the signal to the level suitable for the ADC. The common part of every receiver architecture apart from fully digital ones is that the antenna feeds signal through an RF Band Pass Filter (BPF) to a Low Noise Amplifier (LNA). Automatic Gain Control (AGC) keeps the signal level compatible with the ADC. A main objective during the design of an optimal RF front end is to achieve a suitable dynamic range and minimizing additive noise while minimizing the power consumption. Usually, there has to be trade-off between power consumption and dynamic range.

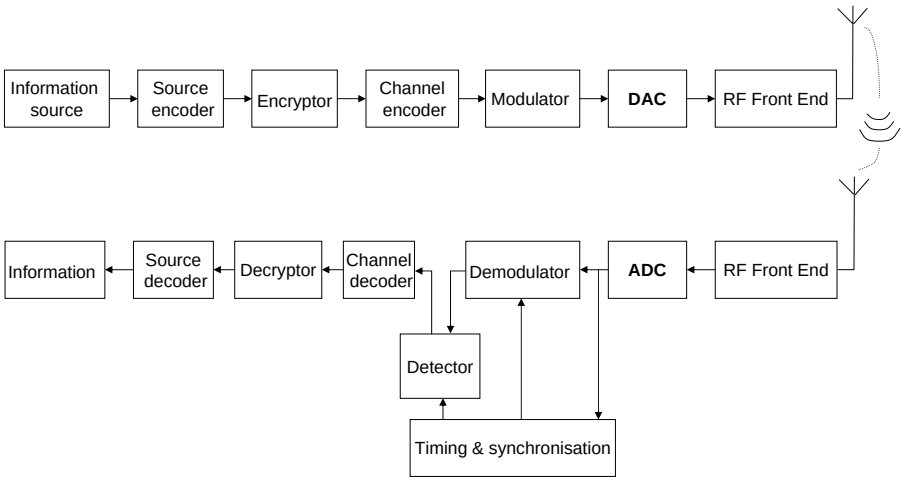

Figure 1.3: Block diagram of a digital radio system

1.3.2 The classical superheterodyne architecture

The superheterodyne architecture has become an obvious choice in receivers since its invention by Armstrong in 1917. Fig. 1.4 shows the classical superheterodyne architecture. In this architecture, the received signal is translated into a fixed Intermediate Frequency (IF) that is lower than the center of the RF signal. The frequency translation is done in two stages because of many advantages of such an architecture: it has lower filtering and quality factor requirements and relaxes the need for isolation between the mixer inputs and the Local Oscillator (LO). The bandwidth or center frequency of the superheterodyne's filters cannot be changed. They are designed according to specific standards that makes this architecture unsuitable for the wideband RF front end of a SR system.

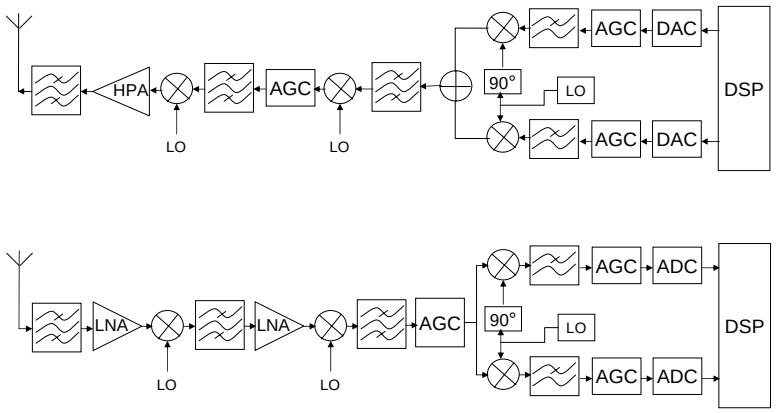

Figure 1.4: Superheterodyne transmitter/receiver

1.3.3 Software radio architecture

A SR architecture is shown in the Fig. 1.5a. SR intends to put the converters as close as possible to the antenna and then compute the radio signal processing in a processor as shown in Fig. 1.5a. A more realistic SR architecture is given in Fig.1.5b. In this architecture the digital part of the transceiver is placed as close to the antenna as possible but some RF front end is indispensable, at least for amplifying and filtering.
The need for this software radio architecture raises a number of technical challenges, which play a significant role in the development of future personal communication systems generation. Most important of these technical challenges are related to analog to digital conversion (ADC) and digital to analog conversion (DAC), which in many cases cannot provide the advanced hardware features needed to support the demanding telecommunication services.

Then, the major problem that the SR technology faces is that the actual ADCs are not able to cope with that very high frequency signals. All signals need to be converted to the digital domain, otherwise digital processing makes no sense. The ADC and DAC are among the key components [2, 86] for SR equipments. In many cases, they define the bandwidth, the dynamic range and have a deep impact on the power consumption of the radio. The wideband ADC is one of the most challenging tasks in SR design. The wideband and the dynamic range of the analog signal has to be compatible with the ADC.

The physical upper bound for capabilities of ADCs can be derived from Heisenberg's uncertainty principle. For instance, at 1 GHz bandwidth the upper limit for dynamic range

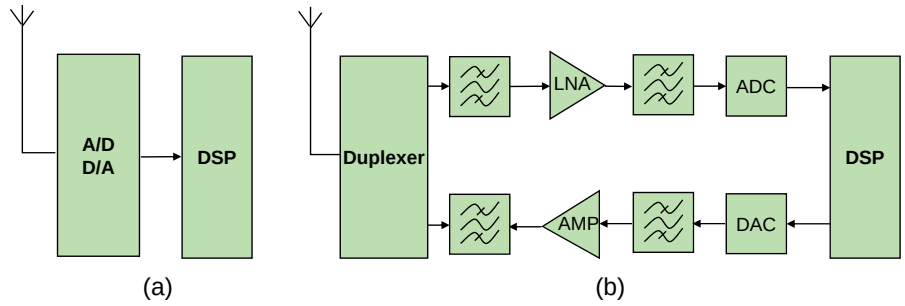

Figure 1.5: a) SR transceiver architecture b) A more realistic SR architecture

is 20 bits or 120 dB [56]). However there are other limiting factors including aperture jitter[(1)] and thermal effects. Unfortunately, the advances in ADC performances are very slow, unlike in many other technologies related to SR. The number of bits in the ADC defines the upper limit for achievable dynamic range. A higher dynamic range requires a higher stop-band filter attenuation. For instance, a 16-bit ADC needs over 100 dB attenuation in order to reduce the power of aliased signal under the half of the energy of the Least Significant Bit (LSB) [56].

The state of the art ADCs for wireless devices sample at rates of nearly 100 Million Samples Per Second (MSPS), quantize the signal with 14-bit resolution and 100 dB Spurious Free Dynamic Range (SFDR[(2)]). A large SFDR is needed to allow recovery of small scale signal when strong interferences are present. These performances do not fulfill the desired level of the required dynamic range mainly when the ADCs have to cope with signals of large bandwidth and high dynamic range. Commercially there is already a demand for even better converters for base stations [14]. Still, it should be mentioned that beside the dynamic range, the sample rate has to fulfill the Nyquist criterion. Higher frequencies cause aliasing and therefore the ADC should be preceded by an anti-aliasing filter.

The relationships that exist between some basic parameters of an ADC are illustrated at an abstract level in Fig 1.6. Simultaneously working towards a high bandwidth and wide dynamic range results in increased power consumption and cost, which are undesirable. These considerations lead us to conclude that, the bandwidth, the ADC has to digitize must be reduced. The solution would be provided by software defined radio's (SDR) architectures as shown in the following subsections.

1.3.4 Direct conversion architecture

The direct conversion architecture needs lower number of parts and it is conceptually attractive due to its simplicity. Its main advantage is that there are no need for any translation into IF. In the receiver side, the signal is directly down converted to baseband. The

[(1)] APERTURE JITTER is the variation in aperture delay from sample to sample. Aperture jitter shows up as input noise

[(2)] SFDR is a measure of the relative size of the largest harmonic with respect to the fundamental for a defined range of pure sine-wave input frequencies.

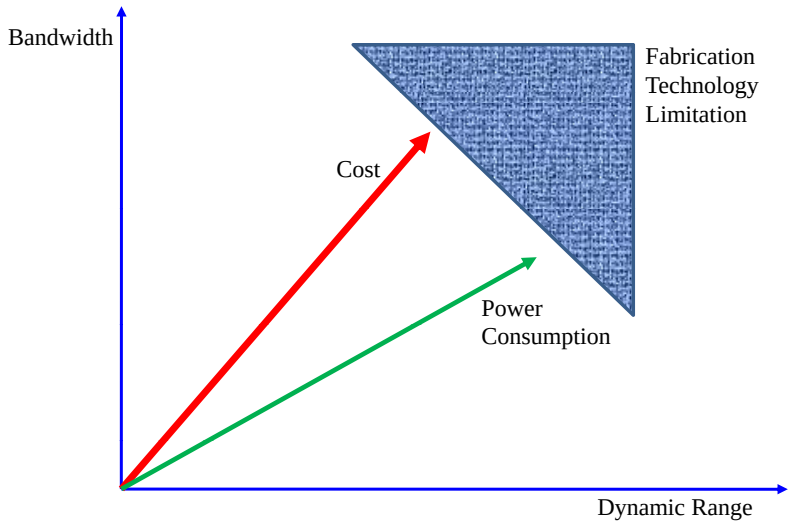

Figure 1.6: Trade-offs and limitations of converter performance

down converted signal is then prefiltered by a variable frequency anti-aliasing filter and, after analog-to-digital conversion, desired channels are chosen by software filters. Fig. 1.7 illustrates the transmitter-receiver architecture. Direct conversion has been so far suitable only for modulation methods that do not have significant part of signal energy near DC (Direct Current). Despite its advantages, this architecture presents two main drawbacks. The first one is associated with the fact that the LO is at the signal band which poses possible unauthorized emissions and internal interference. Thus, this architecture needs an extremely stable LO. This problem can be compensated with digital post processing. The second inconvenience comes from the fact that this LO is not capable to synthesize all the carrier frequencies of the different standards the receiver has to support. Although these inconveniences, the direct conversion architecture was suggested as promising architecture for the future SDR systems [87] since it can offer the possibility to switch between some specific bands.

1.3.5 Sub-sampling architecture

A simplified diagram of a sub-sampling receiver architecture is shown in Fig. 1.8. A sub-sampling receiver is different than an ideal SR in that the sampling frequency can be much lower than the incoming RF frequency. The channel select filtering of the sub-sampling receiver is done mostly in the digital baseband processor, so it does provide some of the flexibility desired from an SDR. In practice a completely flexible receiver is very difficult to design particularly if low power consumption is required.

In a sub-sampling receiver architecture as shown in Fig. 1.8, the incoming RF signal is down-converted to baseband using a sample and hold circuit (sometimes referred to as a subsampling mixer or simply sampler). The sampler is clocked at a lower frequency f_s than twice the maximum of incoming RF signal, but higher than the bandwidth of the

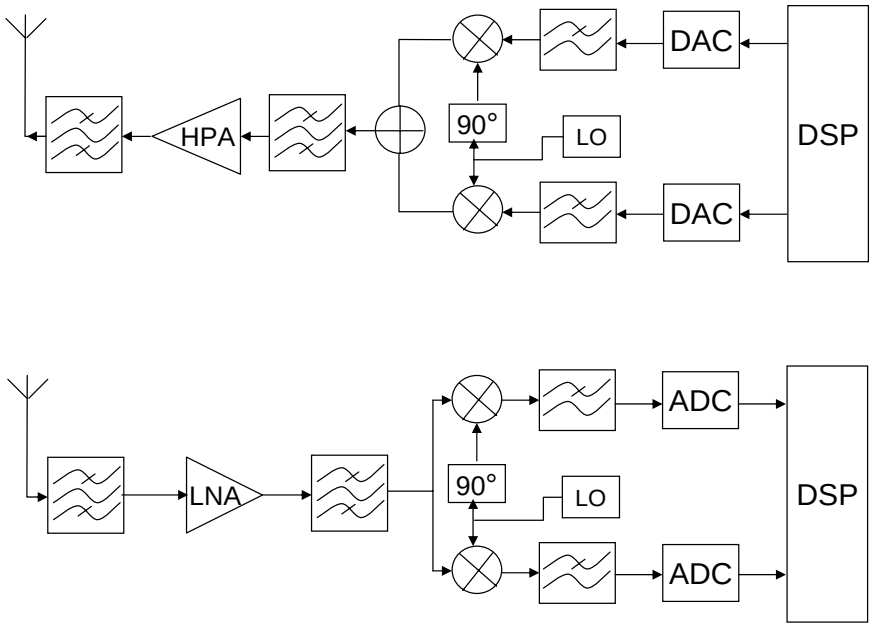

Figure 1.7: Direct conversion architecture

modulated information signal. This type of a receiver can be thought of as similar to the superheterodyne receiver. The first mixing stage is replaced with the subsampling stage and the second mixing it done directly to baseband, but in the digital domain. In a subsampling system, the sampling frequency is usually much lower than the RF input frequency. This requires some appropriate analog filtering in order to avoid aliasing with other signals of the spectrum. The nature of the sub-sampling receiver results in a different architectural design than the more traditional architectures. Sub-sampling receiver architecture is perceived to have limitations due to high noise figure and up-conversion of phase noise [88]. A major drawback of this architecture is that, although, the ADC may just have a low sampling frequency compared to the carrier frequency but it must have an analog bandwidth at least as high as the highest frequency of the signal.

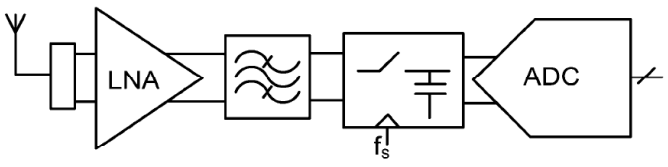

Figure 1.8: Simplified sub-sampling receiver architecture

1.3.6 Feasible SDR architecture

Under the strong constraints discussed above, many publications [68, 14, 89] claim that to cover all services to be supported by the software radio terminal, a limited bandwidth has to be selected out of the full band by means of analog conversion and IF filtering. This concept leads to feasible SDR architecture sketched in Fig.1.9, where the ADC splits the communication chain into two parts: the AFE and the Digital Front End (DFE).

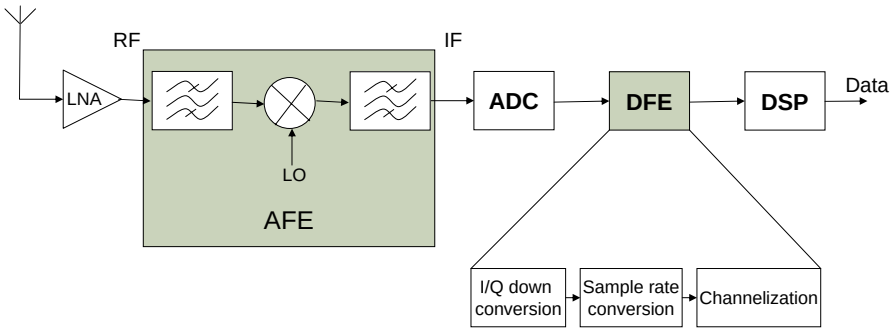

Figure 1.9: The feasible software radio receiver architecture

The AFE selects a bundle of channels and shifts their bandwidth from RF to an IF with which the ADC has to cope with. The DFE is a part of the receiver realizing front-end functionalities digitally that were formerly realized by means of analog signal processing (i.e., down conversion, channelization and sample rate conversion). Channelization comprises of all tasks necessary to select the channel of interest. This includes conversion to baseband, channel filtering, and possibly despreading. Sample rate conversion is a functionality that comes from the statement that it is appropriate to sample the analog signal at a fixed rate. This simplifies clock generation for the ADC, which would otherwise be parameterizable. However, generally signals are often processed at multiples of symbols or chip rates dictated by the different standards. Both facts lead to the necessity to digitally convert the digitization rate to the rate of the current standard of operation.

The conclusion we can draw from the above discussion is that the SR system able to be realized today is a system at half digitized and the second half cannot be digitized before several years at low cost and low power consumption for most of popular wireless standards. That is, before having the availability of advanced ADCs able to provide an extreme dynamic range and very high sample rate capable to digitize the bandwidth of all services to be supported by the terminal and directly after the antenna. A lot of research is being done on advanced ADCs and more solutions will appear in future. The stringent constraints imposed by SR are relaxed by SDR e.g. it may correspond to a digitization at intermediate frequency (as in Fig. 1.9) or even to baseband. We can continue talking about SDR if the reconfigurability of equipment is exploited. In the next section we discuss various reasons that are paving the path of SR technology.

1.4 Drivers for software radio

Following are the factors that are expected to push wider acceptance of software radio from the point of view of end users, service providers and manufacturers:

- Multi-functional systems (End user): a *multi-services* system which provides different services e.g., telephony, data, video streaming, GPS and soon handheld TV broadcast, ITS etc. With the development of short-range networks like Bluetooth and IEEE 802.11, it is now possible to enhance the services of a radio by leveraging other devices that provide complementary services. For instance, a Bluetooth-enabled fax machine may be able to send a fax to a nearby laptop computer equipped with a software radio that supports the Bluetooth interface. Software radio's reconfiguration capability should enable to support an almost infinite variety of service capabilities in a system.

- Global mobility (End user): A number of communication standards exist today and a *multi-standards* system is needed that can support more than one standard. In the 2G alone, there are IS-136, GSM, IS-95/CDMA1, and many other, less well known standards. The 3G technology tried to harmonize all the standards. However, there are many standards under the 3G umbrella. Multi-standard systems can work within one standard family (e.g., UTRA-FDD, UTRA-TDD for UMTS) or across different networks (e.g. DECT, GSM, UMTS, WLAN). The need for transparency, i.e., the ability of radios to operate with some, preferably all, of these standards in different geographical regions of the world is ever increasing. In addition the need for a *multi-band* system which is supporting more than one frequency band used by a wireless standard (e.g., GSM 900, GSM 1800, GSM 1900), and a *multichannel* system that supports two or more independent transmission and reception channels at the same time has cherished the growth of the software radio concept.

- Compactness and power efficiency (Manufacturer): Multi-function, multi-mode radios designed using the *Velcro* approach of including separate silicon for each system can become bulky and inefficient as the number of systems increases. The software radio approach, however, results in a compact and, in the long term, a power-efficient design, especially as the number of systems increases, since the same piece of hardware is reused to implement multiple systems and interfaces.

- Ease of manufacture (Manufacturer): RF components are notoriously hard to standardize and may have varying performance characteristics. Optimization of the components in terms of performance may take a few years and thereby delay product introduction. In general, digitization of the signal early in the receiver chain can result in a design that incorporates significantly fewer parts, meaning a reduced inventory for the manufacturer.

- Easy Infrastructure upgrade (Service provider): In the course of deployment, current services may need to be updated or new services may have to be introduced. Such enhancements have to be made without disrupting the operation of the current infrastructure. A flexible architecture allows for improvements and additional functionality without the expense of recalling all the units or replacing the user terminals. Vocoder technology, for example, is constantly improving to offer higher

quality voice at lower bit rates. As new vocoders are developed, they can be quickly fielded in software radio systems. Furthermore, as new devices are integrated into existing infrastructures, software radio allows the new devices to interface seamlessly, from the air-interface all the way to the application, with the legacy network.

1.5 Research projects in software radio design technology

In this section we review some principal International, European and French SDR projects that contributed to the evolution of SDR from 1990 up till today.

1.5.1 SPEAKeasy

SPEAKeasy, a joint project of the branches of the U.S. military, was one of the first attempts to create a formalized software radio architecture [1]. Research was originally initiated as part of the Air Force's Tactical AntiJam Programmable Signal Processor (TAJPSP), which sought to create a programmable modem that would support the future evolution of waveforms. This project expanded the original goals of TAJPSP to the creation of a software-defined multi-band multi-mode radio (MBMMR) [90]. The SPEAKeasy program was broken into Phase-I and Phase-II.

1.5.1.1 SPEAKeasy phase-I

SPEAKeasy Phase-I was researched from 1992 to 1995. This phase was intended to show that software radio had the potential to [2, 59]:

- alleviate the military's interoperability issues
- simplify the process of incorporating new technology
- provide more advanced security functions,
- simplify information security (INFOSEC) implementations
- add f1exibility without significantly increasing power consumption,
- demonstrate that the use of reprogrammable hardware in the place of dedicated hardware would incur no major performance penalties.

The SPEAKeasy Phase-I system was successfully demonstrated in June 1995. Multi-mode capabilities were demonstrated by interfacing with existing radios employing several different protocols. Programmability was demonstrated by altering standard waveforms on two SPEAKeasy units. The radio was defined to operate from 2 MHz to 2 GHz. An important software radio architecture concept-the division of a wide frequency range into selectable smaller sub-bands was used to simplify RF implementation.

1.5.1.2 SPEAKeasy phase-II

Phase-II sought to design successively more refined software radios and to formalize an architecture. The primary goals of Phase II architecture were to [2, 91]:

- implement an open architecture
- implement a reconfigurable architecture
- achieve cross-channel connectivity
- incorporate reconfigurable hardware

Phase-II designed and implemented a complete software radio architecture based upon a modular design, utilized an open software architecture, and incorporated reconfigurable hardware [92].

After the successful completion of the SPEAKeasy program, the focus was shifted to the Programmable Modular Communications Systems (PMCS) research program. The developments of the PMCS program [3], particularly its entity reference model, were used to form a basis for the U.S. military's continuing software radio research in JTRS. It was also used to help develop U.S. allied software radio projects, particularly in the U.K., Germany, and France, by serving as a basis for the Future Multi-band Multi-waveform Modular Tactical Radio (FM3TR) project and as part of the Atlantic Paw program. To transition the technology into the commercial market, SPEAKeasy technology was also presented to the Modular Multifunction Information Transfer System (MMITS) Forum, now known as the SDR Forum.

1.5.2 JTRS

Joint Tactical Radio System (JTRS) is the U.S. military's ongoing software radio architecture program. Evolving from SPEAKeasy and PMCS, JTRS envisions a radio that aims [3, 4]:

- support operating frequencies ranging from 2 MHz to 2 GHz
- be reconfigurable through waveform software
- support voice, video, and data applications
- be scalable in both software and hardware
- be interoperable with different waveforms, with legacy equipment, and with radios designed for different domains.

For the JTRS program, five unique domains have been identified; airborne, fixed/ maritime, vehicular, dismounted, and hand-held [93, 94]. Rather than impractically trying to construct a single *one-size-fits-all* radio able to meet the demands of each of these domains, JTRS represents a family of software radios. Each radio is designed to address the challenges of its domain. However, each JTRS radio uses the same baseline architecture to ensure interoperability across domains [95]. The baseline (parent) architecture of the

JTRS family is the Software Communication Architecture (SCA) ([5]. The SCA is being built around two key concepts: an open systems architecture and the extensive use of object-oriented methodologies.

The JTRS program is ensuring that its SCA will be able to operate as a domain independent software radio architecture through its validation process and through interaction with software radio standards bodies. By defining itself through an SCA, the JTRS software radio model continues the software radio process of shifting away from a hardware dependent radio architecture to a software-defined radio architecture.

1.5.3 Digital modular radio

The Digital Modular Radio (DMR), is the first software defined radio to have become a communications system standard for the U.S. Military. The compact, multi-channel DMR provides multiple waveforms and multi-level information security for voice and data communications from the core of the network to the tactical edge [4].

DMR currently operate aboard U.S. Navy surface and subsurface vessels, fixed-sites and other Department of Defense communication platforms using frequencies ranging from 2 MHz to 2 GHz. Certified to pass secure voice and data at Multiple Independent Levels of Security (MILS) over HF, VHF, UHF, and SATCOM channels, the DMR system was developed to the U.S. Navy's specifications and meets all the stringent environmental, EMI and performance requirements for use in the U.S. Fleet.

1.5.4 Wireless information transfer system

The Wireless Information Transfer System (WITS) [96] evolved from Motorola's participation in the SPEAKeasy project and later in the JTRS program and the Navy's DMR program. The WITS radio is the first instantiation of the JTRS/SDR Forum architecture. Today, as a JTRS-compliant and SDR Forum-based architecture, WITS-based systems are used extensively by the Navy and are migrating to other government agencies and the broad commercial market.

The WITS radio achieves its high degree of flexibility by creating a software defined radio based upon a dynamically linkable object-oriented design methodology supported by well-defined layers and interfaces coupled with an intelligent hardware design.

1.5.5 CHARIOT

The Changeable Advanced Radio for Inter-Operable Telecommunications (CHARIOT) software radio was developed at Virginia Tech. CHARIOT is most valuable in the mobile/handset domain. In this domain, it is particularly challenging to include enough processing power using only DSPs and microprocessors to support the high data rates of modern systems within the required form factor. Previously, these requirements had required an ASIC solution, which severely limits the flexibility of radio, typically preventing the implementation of a software radio in a handset.

CHARIOT sought to solve this problem by creating a formalized structure to implement a software radio using reconfigurable hardware e.g., FPGAs and PLDs, in a runtime environment. CHARIOT's architecture provides a way for small form factor radios to perform complex data processing at high data rates with efficient resource allocation while maintaining the reconfigurability, flexibility, and scalability expected of a software radio [2]. CHARIOT's Layered Radio Architecture (LRA) was created to address these goals. This LRA leverages the concepts of layers, stream-based processing, custom computing machines (CCMs), and hardware paging.

1.5.6 SpectrumWare

SpectrumWare project began at MIT in the mid-1990s to investigate the suitability of general-purpose processors (GPPs) for implementing software radios. This project sought to leverage the rapid advances in performance and cost-reductions of general-purpose processors to software radio applications. It illustrates the strengths and weakness of applying general purpose processors to software radios while also supporting applications for the radio.

The project produced some innovative approaches to software design to allow for reusing code, adapting the physical layer, and reducing overall processing cycles. The approach to buffering I/O to circumvent processing jitter inherent to non-real-time OSs and the application level guarantee of processing rate using pulling pipelines are some key innovations introduced by Spectrum Ware. The SpectrumWare approach has been successfully applied to AMPS, GSM, and wireless networking interfacing [6]. The feasibility of this approach will continue to improve as general-purpose computers become more powerful and less expensive over time.

1.5.7 GNU radio

GNU Radio is a free software development toolkit (reasonably hardware-independent) that provides the signal processing runtime and processing blocks to implement software radios using readily-available, low-cost external RF hardware and commodity processors [7]. It is widely used in hobbyist, academic and commercial environments to support wireless communications research as well as to implement real-world radio systems.

GNU Radio applications are primarily written using the Python programming language, while the supplied, performance-critical signal processing path is implemented in C++ using processor floating point extensions where available. Thus, the developer is able to implement real-time, high-throughput radio systems in a simple-to-use, rapid-application-development environment. While not primarily a simulation tool, GNU Radio does support development of signal processing algorithms using prerecorded or generated data, avoiding the need for actual RF hardware.

In addition to these projects, some forums involving governmental and industrial actors play important roles in SDR activities. The most active one is the **SDR Forum**.

1.5.8 SDR forum

The SDR Forum is an independent organization comprised of member companies and associations with experience and/or interest in software radio related issues. The SDR Forum is attempting to address the diverse interests associated with software radios and to formulate a standard for use in all future software radios [97]. By utilizing a standard, the SDR Forum hopes to foster the growth of software radio technology and the creation of an open and diverse commercial market.

SDR Forum's architecture has closely paralleled the development of the JTRS architecture. The SDR Forum is expanding on the JTRS's SCA and is examining how different structures will fit into the SDR Forum's architecture. The goals of the SDR Forum architecture are to:

- implement an open systems architecture,
- be domain independent across all commercial, civil, and military implementations
- support all existing waveforms, i.e., commercial, civil, and military
- support the insertion of future technology

The SDR Forum's architecture is primarily focused on the creation of a standard that can be applied to all implementations and used as a measuring stick to determine architectural compliance.

1.5.9 European and French SDR projects

European Commission funded research activities that are mainly implemented within two frame work programs: ACTS (Advanced Communications Technologies and Services) and IST (Information Society Technologies). A list of the main projects is provided in section A.1 of Appendix A. In this section we describe, End-to-End Reconfigurability (E^2R) which is the major European project in SDR.

E^2R

The End-to-End Reconfigurability (E^2R) is one of the major European projects that are influencing the structure of the wireless industry today. E^2R is composed of 32 partners (Motorola, Alcatel, France Telecom, CEA, Supélec, Eurecom ...). E^2R aims at realizing the full benefits of the diversity within the radio eco-space, composed of wide range of systems such as cellular, fixed, wireless local area and broadcast [98]. The key objective of the E^2R project is to devise, develop, trial and showcase architectural designs of adaptive communication systems to offer an extensive set of operational choices to the users, application and service providers, operators, manufacturers, and regulators in the context of heterogeneous systems.

The E^2R Project is structured in eight technical workpackages (WP), corresponding to five main research challenges and three research domain skills [99]. One additional workpackage is dedicated to project management (WP0).

- WP1-E^{2}R Sustainable Business Development and Project Exploitation
- WP2-End-to-End Reconfiguration Management and Control Architecture
- WP3-Efficiency Enhancements for Radio Resource and Spectrum
- WP4-Unified Robust Reconfigurable Connectivity
- WP5-E^{2}R European Reference Prototyping Environment
- WP6-Cognitive Networks
- WP7-Reconfigurable Equipment
- WP8-Proof-of-concept

E^{2}R-II will contribute to the realization of the ambient intelligence vision [100] through which a modern society will interact and communicates with key capabilities of the eco-system and finally actively influence European industrial and economic competitiveness. Standardization, dissemination and knowledge sharing under training instruments will be intensely used to consolidate this influence.

E^3

The End-to-End Efficiency (E^3) project, a continuation of E^{2}R, aims at integrating cognitive wireless systems in the Beyond 3G (B3G) world, evolving current heterogeneous wireless system infrastructures into an integrated, scalable and efficiently managed B3G cognitive system framework. The key objective of the E^3 project is to design, develop, prototype and showcase solutions to guarantee interoperability, flexibility and scalability between existing legacy and future wireless systems, manage the overall system complexity, and ensure convergence across access technologies, business domains, regulatory domains and geographical regions.

French SDR Projects

In France, several **RNRT** (Réseau National de Recherche en Télécommunications) projects are funded. RNRT priorities for year 2002 focused among others on high level system design and associated verification tools of the Software Radio architecture implementation on the platform. The main targeted application domains are Software Radio system for 3G and 4G and SoC in particular. A list of some main RNRT projects is given in section A.2 of Appendix A.

1.6 Challenges in designing SDR

In this thesis, we try to tackle the design of SDR equipment as a problem of granularity adjustment. We cut into pieces the radio chain differently in order to take benefits of another dimension in order to more easily reach the final goal of an ideal SDR design tool. Our view of looking into the SDR design problem is an original contribution and it

does not have to do anything with already existing challenges. One of the biggest existing challenges is hardware/software co-design, which is not only related to the SDR design but any real time embedded system designer faces this problem [101, 102, 103, 104, 105].

Because of the complexity increase and mix of hardware/software, researchers are looking for high level design for SDR [10, 106, 107].

Another focus in SDR domain is SCA [5] for military applications. Specific tools are developed and researches are done in this respect. Zeligsoft [12], PrismTech [13, 108]) and Communications Research Center of Canada (CRCC) [11] have developed specific tools based on SCA. For researches we can quote Wireless INTeroperability for SECurity (WINTSEC) project [109] and Thales Communications researches topics.

After this brief introduction of SR technology and the various related architectures and aspects, our studies will focus on the digital part of the receivers and specifically on the physical layer. The SR's problematic, in this work, will be tackled from the viewpoint of a research of commonalities between several standards and in the standards themselves. In this context, an important technique called *parametrisation* that aims to optimize the resources use in the SR system was introduced [15]. This technique will be presented in details in the next chapter.

1.7 Conclusions

In this chapter, we have traced the evolution of software radio technology. We started with the discussion about the need for the SR. We came to the conclusion that to cope with ever increasing demands of users, manufactures and service providers, the focus has been shifted toward SR approach and conventional Velcro approach of designing radio terminals and basestations will become obsolete very soon. The motivation for introducing optimization aspects of flexible radio system design is the tremendous interest in reducing the cost, complexity and size of the radio terminal/base station which are required to support such a wide range of existing and future wireless technologies and services We discussed different receiver architectures and identified the feasible SDR architecture. SDR projects' survey showed how the software radio concept is developing and identifies key concepts and technologies useful for the practical implementation of a software radio. Software radios and their architectures will continue to progress as new technologies become available. Technologies are advancing the state-of-the-art software radio architecture [62], leading to the implementation of an ideal software radio and eventually cognitive radio [110].

Chapter 2

Parametrisation Techniques for Software Radio

Contents

2.1 Introduction

Wireless communications is based upon transmission standards, e.g. the air interface as well as protocol stack of transmission. A standard completely describes how a signal looks on air and which procedures are used to set up, maintain, and to close down a connection. A conventional approach to the implementation of multi-standard radio is the utilization of multiple transceiver chains each dedicated to an individual standard. Such an approach is not flexible, as most of the hardware needs to be replaced whenever the characteristics of the interface change. This conventional approach called *Velcro* approach does not exploit any common aspects between different standards in order to factorize some hardware devices.

In order to exploit to great advantage the commonalities among the various signal processing operations of a multi-standards equipment a technique called *parametrisation* was introduced in [14]. Parametrisation is an important aspect of the SR concept. It consists in identifying all the common aspects of the mobile communication modes and standards

which the receiver is intended to handle and then to make a generic operation, the behaviour can be changed just by changing parameters. Parametrisation can be considered as a sub-part of the digital radio design methodology.

With the tremendous increase in technology development and proliferation, design of a multi-standards equipment having; several wireless access modes (GSM, UMTS, etc.), local area networks (WiFi, WiMAX, etc.), broadcast (DVB-H, DVB-T, DAB, DRM, DMB, etc.), positioning (GPS, Galileo, etc.), ITS etc., is gaining popularity. In this context, parametrisation can be considered as crucial optimization process which can help in the design of reconfigurable equipments.

According to the methodology of parametrisation, the common aspects of the different standards will become one common processing procedure, which could be installed in the device. This common procedure could be executed by a simple *call*, thereby resulting in a gain of both size and time with respect to the amount of code to be downloaded or read in order to modify the radio behavior. From the design point of view, the parametrisation will optimize the co-design and will probably reduce the time to market by enabling incremental design.

To perform parametrisation, a first technique, the *common function* (CF) technique, was introduced in [15, 16]. The author in [15] highlights the fact that the CF technique can be seen as highly dependent upon the standards under consideration. As a remedy to this problem, the common operator technique presented in section 2.2.2, aims to be less standards dependent.

Study presented in [17] can be considered as a foundation stone of CO technique, later complemented by additional CO as described in [18, 19]. Furthermore, in [20, 111], a theoretical approach (TA) and an optimization process are proposed to identify CO in a multi-standard terminal, viewed as a graph. It is precisely the objective of work presented in this thesis.

This chapter is organized as follow: in the first section, we discuss the concept of parametrisation techniques as a methodology to design SR. First, we highlight the importance of parametrisation and then we present the state of the art that lead to the definition of CF technique. Next, the CO technique is introduced and two approaches for designing CO are discussed, i.e., the pragmatic and the theoretical approach. The theoretical approach is the foundation of this thesis. In the next section we apply the common operator technique to two operators i.e. FFT operator and LFSR operator. We present some examples of the usage of these operator. Finally a conclusions' section ends this chapter.

2.2 Fundamentals of the techniques of parametrisation

The initial work of J. Mitola [54] and W. Tuttlebee [14] defined the SR as *a set of techniques that permit the reconfiguration of a communication system without the need to change any hardware system element.* A multi-standards system can be implemented by *Velcro* technique as mentioned earlier. It consists in instantiating all required standard

implementations and enabling reconfiguration by activating the desired one. This technique does not exploit the inter and/or intra standard commonalities. Parametrisation was introduced in order to exploit these commonalities by minimizing the size of the code that permits to change the behaviour just by changing parameters which are 1) fast to transfer inside the equipment, and 2) fast to upload from somewhere else in the network (less bandwidth consuming).

2.2.1 The common function technique

The techniques of parametrisation originated with the proposition of function sharing among different standards. This led to the definition of the *common function* (CF) technique as stated in [14]: *It seems natural to ask whether there are construction principles that are common to all standards and, if so, to describe the standards according to these principles.*

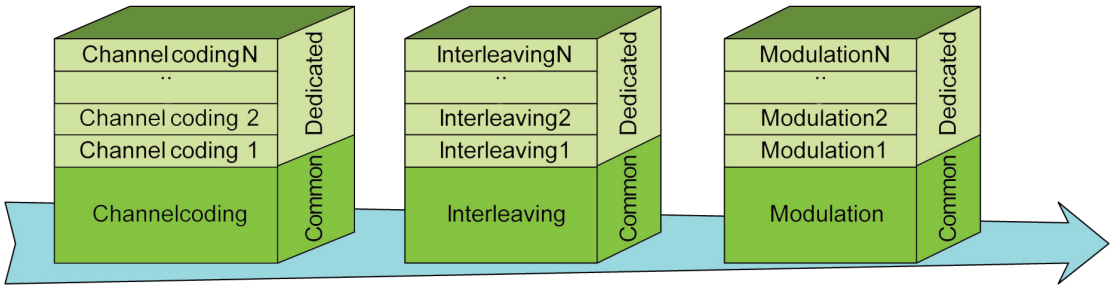

Figure 2.1: Multi-standards transmitter data processing tasks

Let us explain the CF technique. This technique can be illustrated by Fig. 2.1. For each *transmitter data processing tasks*, all the components dedicated to the same functionality were aggregated in the same CF. Fig. 2.1 delineates three CFs, namely; channel coding, interleaving and modulation. Each CF is depicted as the combination of one *common* and *several* dedicated slices. The *common* part includes the components required by at least two functions and each dedicated part is characterized by non-related components of each individual CF.

The resource sharing inferred by the CF technique allows the non-duplication of redundant components and a possible save of complexity (memory size, downloading speed, reconfiguration speed, computation complexity, etc.). From this point and based on the previous definitions, it is suggested in [112] that a parametrisation technique should be a method that searches for and finds all the commonalities between several different standards in order to optimize the resources during the equipment's implementation and/or the execution phases.

Examples

In this section we explain with examples the implications of the parametrisation method. Let us start with the example that focuses on the development of a general parameterizable modulator [15]. Author explains in detail how one can develop a general modulator structure that can process signals for several standards : GSM, IS-136, UTRA-FDD and Digital Enhanced Cordless Telecommunications (DECT). The modulation mode used in GSM is Gaussian Minimum Shift Keying (GMSK). GMSK is a special form of frequency shift keying which is a non linear modulation mode. In contrast to GMSK, the modulation modes Quadrature Phase Shift Keying (QPSK) (used in UMTS) and its derivative π-DQPSK (used in IS-136) are linear. Therefore, the authors in [84] proposed to introduce the linearized version of GMSK into a SDR system. This procedure consists in splitting-up the complex envelope for the GMSK signal into two summands with one representing the linear and the other representing the nonlinear part. Moreover, the linear part contains about 99% of the signal energy. Therefore the GMSK signal may be well approximated by its linear part, i.e. in a software radio the symbols to be transmitted by a GMSK signal may be pulse shaped with an impulse $C_0(t)$ that leads to a linear modulation. Using this linear approximation, GMSK signals can be produced by the same linear I/Q modulator employed for PSK signals [15]. The proposed general modulator is shown in Fig.2.2. This modulator is driven by several parameters and the modulation mode can change by an adjustment of these parameters. More details about the parametrisation and functioning principle of this modulator can be found in [15].

An architecture for SR receiver for GSM and UMTS (UTRA-FDD) is proposed in [113]. In this architecture several signal processing blocks dedicated to an individual standard are duplicated and another is built in such a way to be common to the two considered standards. For instance, there are two different methods used for equalization. In the case of GSM a trellis-based equalizer according to the Maximum A Posteriori Probability (MAP) criterion is employed while for UMTS-FDD a Rake receiver is used, which exploits the advantage of the orthogonality of the different multi-path signals generated by using spreading codes. For channel estimation, both standards use the same function consisting in correlating the known training or pilot sequence with the received sequence. As for the demodulation, the authors explain in [84] the demodulation process for both GSM and UMTS signals. Moreover, two MAP algorithms are implemented in this architecture required for the turbo decoder of UMTS. In order to share common functions the authors proposed to adapt these two algorithms to be used as Viterbi-equalizer and consecutive convolutional decoder in the case of GSM. The received sequence of symbols may be interpreted as the output of convolutional encoder of rate 1 [84]. Therefore for GSM, basically identical algorithms can be used for equalization and for decoding. For UTRA FDD, the data may also be turbo encoded by using a second encoder and an interleaver between the two encoders [114]. The second-MAP algorithm or convolutional decoder can be used by GSM for ordinary convolutional decoding as well as for the UTRA-FDD. In this latest case, the MAP-algorithm should be adapted for the use as a second convolutional decoder in the case of turbo decoding.

As stated in [115], the third generation mobile communication system UMTS will enable a very flexible communication technology which means variable data rates from 8 kbit/s

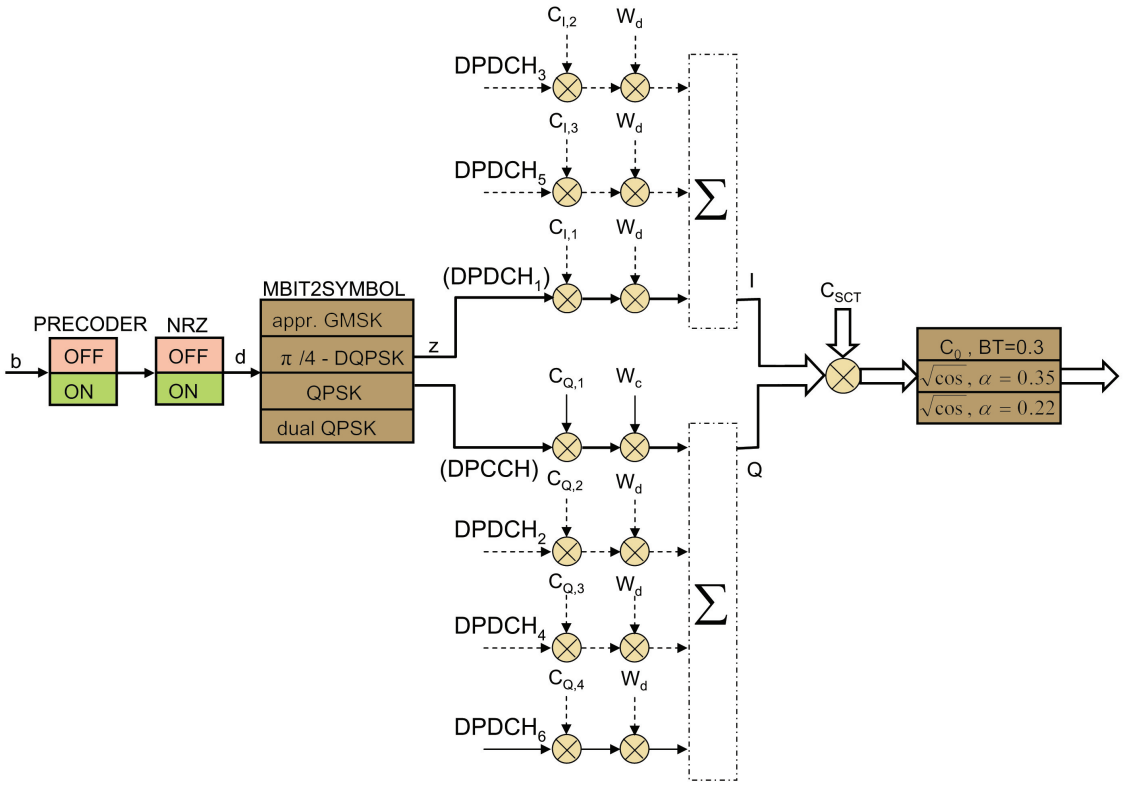

Figure 2.2: Generalized parameterizable modulator [15]

speech up to 2 Mbit/s data transmission. But the new standard can not replace established systems like the European GSM and DECT, instead it is planed as an expansion of the second generation systems. That means future UMTS handhelds must be able to perform second and third generation communication technologies and also seamless system handover. A proposed solution in [115] is to implement all baseband functions like channel coding, modulation, spreading and receiver algorithms in a general, parametrized way, so that all of them can be used for the selected standards. This structure reduces the size of the hardware platform and has a fast performance by changing the air interface for a system handover, because only another set of parameters has to be used. In order to identify common baseband functions the air interfaces of the DECT, GSM and WBCDMA standards have been analyzed.

Another example of the common function approach, for GSM, UMTS and Professional Mobile Radio (PMR), concerning channel coding aspects is given in [16]. As in the previous examples, it is more a matter of an assembly of boxes than a common function. All the boxes are not used at the same time; their use depends on the considered standard. As stated by the author, when a box is not used, the corresponding software is not used and therefore it is not time or CPU consuming.

In [116], a common structure called VITURBO for ordinary convolutional decoder and turbo decoder was proposed. This structure permits to perform the Viterbi and Turbo decoding.

From the CF examples presented above, we can conclude the CF technique depends highly upon the selected standards. The basic design of the CF consists in the aggregation of the required components in only one function with minor sharing. The evolution to new standards must require the addition of the distinct components of each function in the associated CF. As a consequence, the CF should be redefined and redesigned to be capable to meet the requirements of the advanced standards.

Finally, according to the concluding remarks of author in [16] *parametrisation (implemented with Common Function) makes it possible to build communication systems with flexible components, under the restrictive assumption that these components belong to a predefined set of transmission modes.* Using a single processing function, one can cover all standards under consideration.

In consequence of its standards' dependency, the CF technique is considered as a Fixed Technique [112]. From here, comes the idea to proceed toward another approach that will give the possibility to built an open structure. By open structure, we mean a structure whose functionality can be used independently of the processing context or of the communication mode. This new approach called Common Operator approach will be discussed in details in the next section with examples.

2.2.2 The common operator technique

The *common operator* technique follows the principles of the techniques of parametrisation and consists in identifying common elements based on structural aspects. In reference to the definition of the pragmatic approach given in 2.2.4, the intrinsic design of the CO is created independently of any standards. In essence, a CO is used to perform operations without knowing their application and must be defined by a general equation. In practical terms, we want to design CO independent of *calling* functions and as a consequence of typical standards. In contrast to the CF, CO is not specific to the implementation of a terminal; common operator technique is to be considered as an *open technique* [112].

Following all these requirements, it is only appropriate that a common operator could be called and re-called by distinct functions several times all along the terminal. This *reuse* is an important issue in the design and the implementation of the operator. One of the objectives is to execute the common operator in a minimal clock cycle number (analogy with the well known MAC operator in DSP). Let us consider the graphical breakdown of communication chains of a multi-standards terminal as shown in Fig. 2.3 proposed in [20]. This Figure shows that a communication chain is composed of different processing blocks, e.g. source encoding, encryption, channel encoding, modulation and some functions performed by the RF front end. The functionality of each of these processing blocks is achieved by lower level processing operations. From top to bottom, we decrease the granularity level of the considered components up to basic LUT or MAC or even to the

level of primitive operators e.g. adders, multipliers, etc.

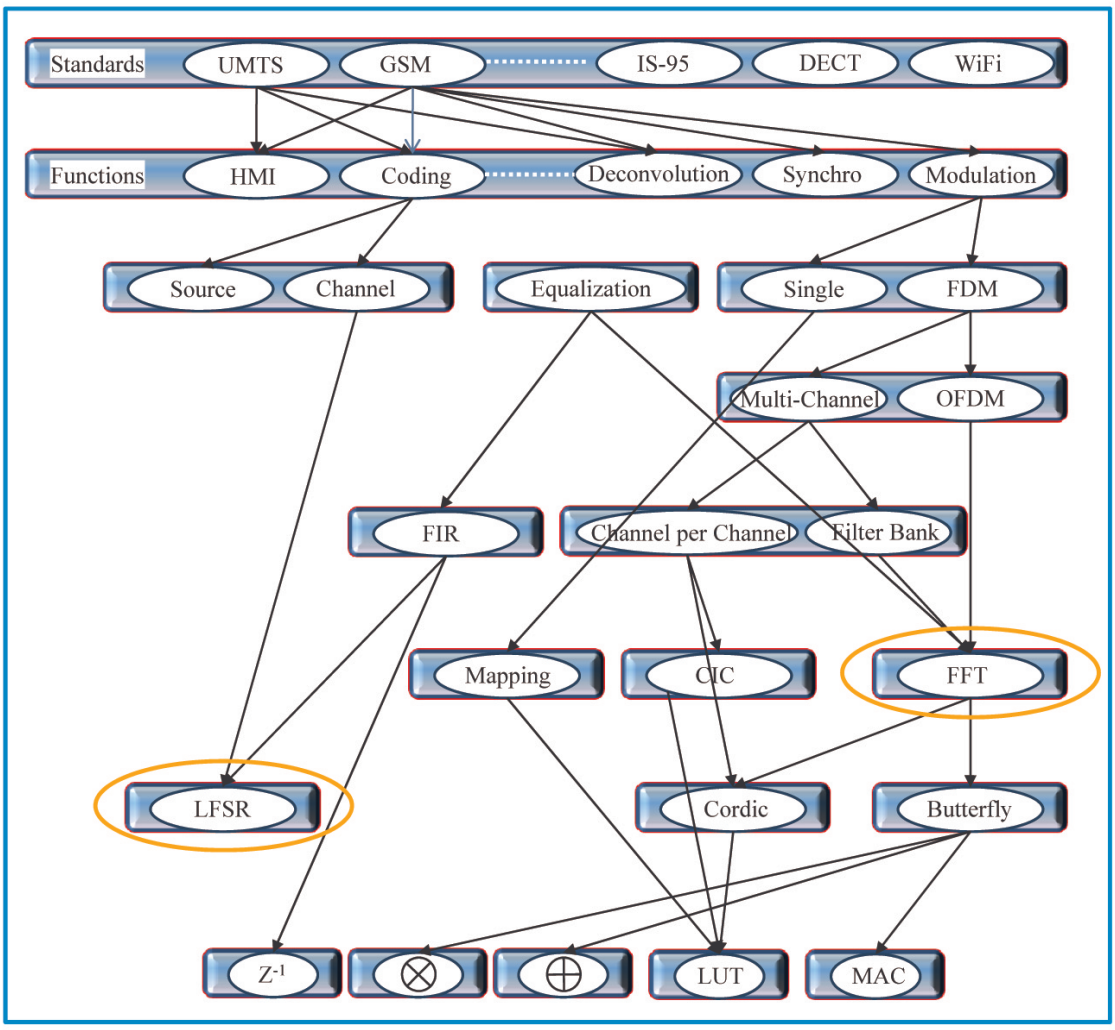

Figure 2.3: Generalized block diagram showing the breakdown of several standards

Although the goal is to find the maximum of common elements and then share their functionalities between several processing tasks, this research becomes useless and ineffective when the latency of systems exceeds certain limitations [117]. So, the optimization in terms of hardware or software resources is directly dependent on the performance in terms of delay or execution time. In order to attain the best cost-performance trade-off, one should identify a level of granularity at which the designer will be based to implement the processing elements of communication standards. Hence, the CO technique consists in increasing the granularity of the basic primitive levels elements to address in an SDR. This operator could be used by as many future standards as possible. This operator has to be completely and successfully checked, optimized and tested. It is expected that the higher the level of the operator is, the shorter the development time of new functionalities will

be. As the goal is to achieve the best compromise between performance and complexity, it is important to determine the ideal level of granularity that designers will choose to design multi-standard equipments, neither the Velcro approach that will of high complexity (but highly parallel), nor the arithmetic operators that will be very sequential (but will have lower complexity).

A consequence of CO technique is a scheduling issue at run time as common parts are reused several times. The scheduling issue for inter/intra standard cases has partially been addressed in [118]. However, in this thesis we do not go into details of scheduling issue and it may be addressed in future by other Ph.D students. In the following section we explain the use of FFT and LFSRs as COs that are shown encircled in the Fig. 2.3.

2.2.3 Comparison of the two techniques

According to the previous discussion about CF and CO, both techniques converge to the same objective: a reconfigurable architecture, which becomes optimal in the sense of the theoretical approach. That means finding paths in the TA, which decrease some costs or which provide a cost/efficiency trade-offs. These two technical methods are very close. The main difference between them consists in the common blocks considered. The CF focuses on similarities between functions whereas the CO looks for architectural similarities inside the functions. Obviously the CO technique is identified to be at a lower level of granularity than the CF technique.

Both CO and CF techniques could be carried out in a hardware manner (by instance on FPGA or even on ASIC) in order to optimize their occupancy, execution rate, general use, etc. As presented in Fig. 2.4, depending on the CF or CO standpoint we can address either the common function technique or the common operator technique.

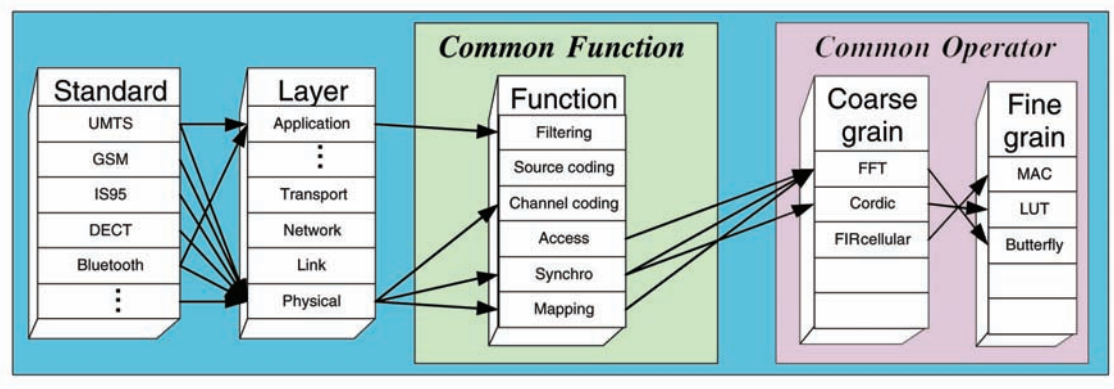

Figure 2.4: Two techniques of parametrisation

Let us conclude the discussion about the two techniques by the following remarks: The CF Technique consists in seeking an optimized generic function (the expected common function) like coding, mapping, etc. which can replace the initial task present in a prede-

fined set of standards (NB: a CF should be not considered in this thesis as a mathematical function). It is a fixed technique [112]. The CO technique claims to be independent of the standards by finding the smallest set of highest-level operators like MAC, FFT, etc., which are used by the maximum number of functions.

A trade-off could be proposed between CF and CO, with the following rule: 1) a CF can call (one or several) functions, 2) a CF can use (one or several) operators and 3) a CO can not call a function. In the next section we discuss the techniques to design COs.

2.2.4 Approaches to design common operators

In the context of techniques of parametrisation, the approaches to identify/design common operators can be divided into two categories:

- Pragmatic approach
- Theoretical approach

These two approaches are distinct but convergent in nature to specify the COs. In this thesis, we are basically concerned with the theoretical approach to specify the COs for multi-standard equipments.

2.2.4.1 Pragmatic approach

The *pragmatic approach* (PA) is the initial approach developed to identify or create possible COs. The approach follows two stages; an Existing Search and a Constructive Search. The Existing Search consists in identifying in literature and in existing works, some operators potentially common. The main interest is to enlarge the classical operations available with the operator considered to non-expected operations and define a new CO.

The Constructive Search is basically the next step in the design of the CO, following the previous Existing Search. It consists in the handmade building of CO. The approach tries to transpose like-looking architectures into a drawn up general block and step by step, merge them into Common Operators of Higher Levels.

2.2.4.2 Theoretical approach

The *theoretical approach* (TA) consists in moving the classical breaking down of Fig. 2.3 into a graphical approach. In the context of *graph optimization problem*, the TA applies an optimization process in order to find the most relevant CO (or combination of CO) present in the graph (explanation of this with examples is given in Part-II). The optimization process related to the TA is performed by a specific cost function to be minimized. The definition of this cost function is given in Part-II of this thesis.

Most of the systems we considered here are standard based. This means, their air interface as well as all their protocols are completely known. Using this standard description of Fig. 2.3, we are able to draw some of the possible paths between the functions so as to perform the required standard.

Based on the scheme introduced by TA, let us give another vision of the parametrisation: *the parametrisation techniques are the means to find the optimal path in the tree derived from Fig. 2.3.* Here we are considering a global optimality. This means that even if a path is not locally optimal for one function, but it contributes to the global optimality, then we consider it. Various criteria for measuring this optimality e.g. memory size, downloading speed, reconfiguration speed, computation complexity, etc. are mentioned in [111].

From the point of view of above definition, three challenges arise:

1. The definition of the graph
2. The definition and development of the cost function
3. The optimization algorithms

These issues are addressed in the forth coming chapters in greater details. The development of graph and its related issues are discussed in chapter 3. The cost parameters, types of the costs and the cost function is described in chapter 4. Different optimization algorithms are addressed in chapter 5.

In a nutshell, both approaches have the same objectives i.e. find the best set of COs. The *theoretical approach* selects the best set among several operators, already in the graph. As a consequence the approach is circumscribed by the quantity and the relevance of the operator present in the graph. The *pragmatic approach* allows us to identify and create possible CO, without necessarily certifying their relevance. Actually, the PA improves the graph of the TA and the TA validates (or not) the CO created by the PA. As a result, once a CO obtained by the PA, it can be integrated in the TA to find out the best optimization. It is worth mentioning here that common functions which, back at that time [113, 115], were considered out of a pragmatic point of view on mobile communication standards. However in our work we back the pragmatism by the theoretical approach as explained earlier.

2.3 Examples of COs

In this section, as an illustration, we propose and describe briefly two different common operators: the Fast Fourier Transform (FFT) operator and the Reconfigurable Linear Feedback Shift Register (R-LFSR) operator, which are results of the Existing Search approach presented in 2.2.4.

We first identify the well known FFT operator as a fair common operator with a large range of practical applications [17, 43]. Next, we focus on the general Linear Feedback Shift Registers architecture and point out that a large diversity of operations could be executed by a LFSR [119].

There are many ways of designing a FFT and LFSR structures. In this chapter, we describe only the existing architectures called FFT and R-LFSR. Afterwards, following the requirement of the Constructive Search we present a new FFT operator called DMFFT operator and Extended Reconfigurable LFSR operator in details in Part-III of this thesis.

2.3.1 FFT operator

The Discrete Fourier Transform (DFT), or its special and attractive class FFT, is extremely important in the area of frequency (spectrum) analysis as it takes a discrete signal in the time domain and transforms that signal into its discrete frequency domain representation. This area in digital signal processing was started with the publication of the Cooley-Tukey algorithm [42] which allows to reduce the order of complexity of some crucial computational tasks in FFT computation e.g. convolution from N^2 to $N\ log_2 N$, where N is the transform length. This processing technique was rapidly credited by the tremendous increase of interest in Digital Signal Processors (DSP) beginning in the seventies and continued to play a key role in the widespread use of digital signal processing in a variety of applications like telecommunications, medical electronics, seismic processing, radars, etc.

This FFT operator, identified as a common element, can be allotted during a given time slot to each function that necessitates the use of Fourier transform, which is exactly what the concept of common operator is about. Then, we can define a CO as an operator that, independently of the application context, can be reused by each function or processing requiring its functionality. However, this operator needs to be parameterizable by some parameters (for example the size of transform, the word length, etc) to perfectly meet the requirements of each application.

The FFT can be used in the implementation of numerous important tasks of a communication transceiver, e.g. filtering, equalization, channelization, OFDM modulation, etc. as stated in [17]. In addition, FFT can also be used for video processing and other domains that will be more and more merged inside the same equipments as radio systems.

The authors begin their discussion by presenting the filtering function, initially defined as a convolution product, as a product of the signal transform (input of the filter) and the frequency response of the filter. To keep the circular characteristics of the convolution, it is necessary to use specific techniques known as overlap-add and overlap-save techniques.

For the channel equalization, the authors show that any type of equalizer (except the MLSE) can be implemented in Frequency Domain (FD); starting from Fast Least Mean Squares (FLMS) [120] and Unconstrained FLMS (UFLMS) [121] through implementation in FD [122], Quasi Newton algorithm in FD [123] and Decision Feedback Equalizer in FD [124] to new Single-Input Multi-Output (SIMO) semi-blind algorithms [125].

It was also shown in [17], that the channelization, channel estimation, (de)correlation multi-user detection, OFDM (de)modulation, despreading and Rake function can all be performed in FD using FFT. Channelization function is one of the main functions implemented by the Digital Front End as already shown in Fig.1.9 [89]. For channelization two approaches are possible. The first one is the classical per channel approach and the second one is the filterbank channelizer [49, 126], which could be performed, under some assumptions, with transform operators, particularly with the FFT. Having enough said about the relevance to use FFT in a radio equipment, let us present an example showing the benefits of FFT as CO.

Parametrisation examples with the FFT operator

The decomposition of three functions: equalization, multi-channel and OFDM is illustrated by a graph in Fig. 2.3. The implementation of the CO i.e. FFT (in frequency domain) in Fig. 2.3 results in the removal of unnecessary paths and leads to the new graph of Fig. 2.5. Fig. 2.5 focuses on a sub-graph of Fig. 2.3 to clearly present the decrease of path with a basic example. In this diagram, the use of the CO not only causes an outstanding decrease of the number of paths but also eliminates the need for some functions (such as the FIR function). Like in Fig. 2.3, only one part of the possible paths are represented in the scheme of Fig. 2.5.

Although the FD version is not locally optimal (like in the short FIR case), taking into account the fact that the FFT operator is mandatory for multi-carrier modulation, we globally optimize its use with this method. This results in saving of some processing elements and consequently optimize the manufacturing cost of the corresponding multi-standard system. After taking a more detailed look at all these functions, it becomes very clear that, if they are performed in the Frequency Domain, the FFT used could be very different from one function to another. For example, for a short FIR, or for a GSM channelization function, we need a short FFT (typically 128 points), whereas for DVB-T demodulation, the length of the FFT should be 8 K points. Knowing that it is possible to perform large FFT with a short FFT [127, 128], thus, the optimal operator will be a short FFT or may be the basic butterfly.

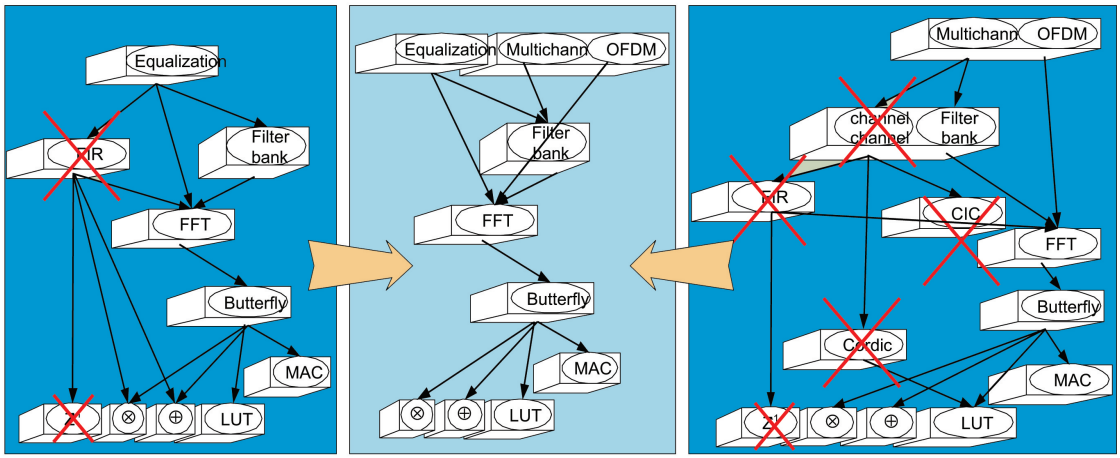

Figure 2.5: Optimal path for the channelization, equalization and OFDM demodulation

2.3.2 LFSR operator

Several architectures called Shift Registers and Linear Feedback Shift Registers can be found in literature that are extensively used in various standards and/or existing transceivers. Let us explore the implementation of a specific LFSR: the Reconfigurable Linear Feedback Shift Register [46]. The R-LFSR can be considered as an interesting common operator for

transceiver functions at the bit level.

The genesis of R-LFSR structures came from the observation that many operations of several ongoing standards derive from Linear Feedback Shift Registers or Shift Registers. These operations mainly concern *pseudo random sequences generators*, *scramblers*, *convolutional coder*, *cyclic redundancy check* and *block channel coding* (extended to Reed-Solomon codes).

The so-considered R-LFSR was designed to substitute these operations as a common operator with specific parameterizations. This operator has been implemented for three standards.

As a consequence, it should be called a large number of times. But in a real implementation it is not sensible to consider only one instance of a R-LFSR cell that would be called by the functions of the standards. Thus is due to data and control dependency which exist between these functions and would lead to inefficient buffering and synchronization overhead. Following the examples of filters banks, the problem has been solved by implementing a bank of Common Operators. A comprehensive discussion about the LFSR COs and their applications with examples is presented in Part-III of this thesis.

2.4 Conclusions

In this chapter we have discussed the common function and common operator techniques in the context of techniques of parametrisation. Starting with some conceptual considerations we can say that the parametrisation is a technique that aims at optimizing the cost-performance trade off while building a multi-standard terminal. These techniques, explained with examples will be further justified in next chapters. Two approaches to identify and develop CO technique have been explained. Theoretical approach involves the identification of an optimal level of granularity, from which a component can be considered as a common communication block enabling its reuse by several applications. The selection of the most appropriate level of granularity helps to balance between economy and computing efficiency. Questions that were being raised during the discussion about the theoretical approach will be answered one by one in the forth coming chapters of Part-II of this thesis.

Part II

Introduction to part II

In this part of thesis, we will discuss our idea of modeling air interface standards as graphs. After drawing these graphs we get a clear picture of different alternative that are available to perform the same task.
Here the main question that comes to our mind is that OK, different alternatives are available but which of these alternatives gives the optimal solution? Which path in the graph is the most efficient? Which of the available alternatives balances between cost and economy? To answer these questions we are left with no other choice but to turn our graph model into an optimization problem. To do this one of the many possible solutions is to associate cost parameters with the functional blocks and arrows of the graph i.e. turn the graphs into weighted hypergraphs.

Finding these cost parameters and then associating them to graph is a challenging problem. We identify some cost parameters that can be associated with graph entities. Now expressing the costs in terms of identified cost parameters, itself is a real hard task as even with a single cost parameter, there are hundreds of different possibilities depending upon platforms, target technologies, clock speeds etc. Hence these costs depend upon the choice of digital hardware e.g. DSP, FPGA and ASIC.

After considering these parameters and associating them with the graph entities (blocks and arrows), our graph is turned into an optimization problem. Here starts another hard task of formulating our objective function. This formulation is discussed in detail in the section 4.4. After having the objective function our next step is to find its solution. This optimization problem can be solved by different techniques. Among these techniques we have the choice to select between the optimal and near optimal solutions. Due to certain limitations, techniques providing optimal solution are no longer feasible and we are left with no choice but to switch to techniques that provide near optimal solution. These limitations are discussed in chapter 5 of this part. Having considered many techniques that give sub-optimal solution we have come to the conclusion that Simulated Annealing and Genetic Algorithms are best suited to the kind of the problem that we have at hand.

This part starts with the chapter 3 in which we explain how to model air interface standards are graphs. Starting from the fundamentals of graph theory and after exploring the already graphs we conclude that for our problem we are to use hypergraph. This is because of the dependencies needed for our problem which were not available for other graph types. After this we describe the development of graph theoretic models for multi-standards SDR systems. Since we have extensive literature available in the domain of network theory so we try to recast our problem as network design problem. Later because of some difficulties we decide not to convert our graph into a network.

In chapter 4 of this part we start our discussion with different cost parameters and types of the costs. Then we discuss the digital hardware composition of SDR. Following this we provide a brief summary of trade-offs in using three foundational digital hardware available: DSPs, FPGAs, and ASICs. Based on the selected costs associated with different graph entities we formulate our cost/objective function to be solved by optimization techniques presented in chapter 5. We formulate our objective function as multi-objective opti-

mization problem (MOP) also known as multi-performance/vector optimization problem. Many real-world scientific and engineering MOPs are irregular and hence deterministic search techniques are not suitable for them. To solve these irregular problems, stochastic search and optimization approaches such as Simulated Annealing, Monte Carlo methods, Tabu search and Evolutionary Computation Algorithms were developed as alternative approaches that are addressed in chapter 5.

In chapter 5 of this part, we first discuss the optimization techniques. We explain the selected optimization techniques that are most suited to our problem. We develop a tool based on these techniques that can be used to solve our problem. We present a design scenario with the costs based on our intuition and solve it by the selected techniques of optimizations implemented by our tool. Further, we use this tool in part III of this thesis where we present real design scenarios using different common operators with real costs and solve the optimization problem with our tool.

Chapter 3

Graph Modeling of SDR Systems

Contents

3.1 Introduction

In the previous chapter we concluded our discussion by saying that our work in this thesis is centered around the common operators' technique of parametrisation and for CO technique we will concentrate on theoretical approach that is based on graphical formalism. In this chapter we address the first question raised in this regard i.e. defining the graph for the TA. We start this chapter with an introduction to the graph theory and we present different types of graphs already available. We mention some of their applications in diverse domains. After that we explain the development of a hypergraph that is well suited to the problem at hand. We explain this with different examples in a step by step manner. Then we explain an other perspective in which we consider the conversion of hypergraph into a network design problem. This aspect may be thought as a result of a search to explore extensive literature available for the network theory. Finally, a conclusions' section ends this chapter.

3.1.1 Basics of graphs

Configurations of nodes and connections occur in a great diversity of applications. They may represent physical networks, such as electrical circuits, roadways, or organic molecules. They are also used in representing less tangible interactions as might occur in eco-systems, sociological relationships, databases, or in the flow of control in a computer program [21].

Graph-theoretical ideas can be traced back to 1735 when Leonhard Euler (1703-83) presented his solution of the Königsberg bridges problem, although the first mention of a graph was not until 1878.

Any mathematical object involving points and connections between them may be called a *graph*. In computer science, the word *graph* is commonly used either to mean a graph as $G = (V, E)$ (defined below) or to mean a computer-represented data structure whose value is a graph. A graph is realized in a plane or in 3-space as a set of points, representing the vertices, and a set of curved or straight line segments, representing the edges. The curvature or length of such a line segment is irrelevant to the meaning. However, if a direction is indicated, that is significant. Let us start with basic terminology of graphs.

Basic Terminology [21]

- A graph $G = (V, E)$ consists of two sets V and E.
- The elements of V are called *vertices* (or *nodes*).
- The elements of E are called *edges*.
- Each edge has a set of one or two vertices associated to it, which are called its *endpoints*. An edge is said to *join* its endpoints.

A line drawing of a graph $G = (V, E)$ is shown in Fig. 3.1. It has vertex-set $V = \{u, v, w, x\}$ and edge-set $E = \{a, b, c, d, e, f\}$. The set $\{a, b\}$ is a multi-edge with endpoints u and v, and edge c is a self-loop.

Simple Graphs: Most of theoretical graph theory is concerned with simple graphs. This is partly because many problems regarding general graphs can be reduced to problems about simple graphs. *A simple graph is a graph that has no self-loops or multi-edges.*

Digraph: If all the connections in a graph are unidirectional, it is called a digraph. An edge between two vertices creates a connection, and it can have two opposite senses. As signing a direction makes one of these senses *forward* and the other *backward*. Viewing direction as an edge attribute is partly motivated by its impact on computer implementations of graph algorithms. A *directed edge* (or *arc*) is said to be directed from its *tail* and directed to its *head* (The tail and the head of a *directed self-loop* are the same vertex.).

Many problems in science and engineering can be modeled in terms of directed and undirected graphs. The data structures and algorithms used to represent graphs can have a significant impact on the size of problems that can be implemented on a computer and the speed with which they can be solved. Moreover, allowing graphs to have additional

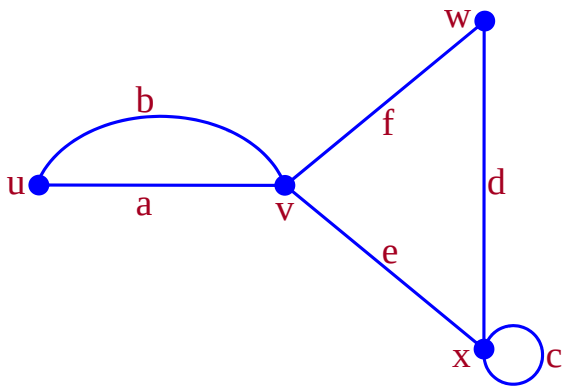

Figure 3.1: A graph

attributes beyond vertices and edges e.g. weights, costs etc. enables them to serve as mathematical models for a wide variety of applications.

Directed Acyclic Graph: When a digraph has no directed cycles, it is called a directed acyclic graph, or a DAG (A *cycle* is a closed path of length at least 1, while *length* is number of edges). While being acyclic may seem to be a stringent condition, it arises quite naturally because vertices often have a natural ordering. For instance, vertices may represent events ordered in time or ordered by hierarchy. This ordering makes results and algorithms for DAGs relatively simple.

Hypergraph: A hypergraph is a generalization of a graph, where edges can connect any number of vertices. Formally, a hypergraph H is a pair $H = (X, E)$ where X is a set of nodes or vertices, and E is a set of non-empty subsets of X called hyper-edges or links. While graph edges are pairs of nodes, hyper-edges are arbitrary sets of nodes, and can therefore contain an arbitrary number of nodes. A hypergraph is also called a set system or a family of sets drawn from the universal set X.

Let us describe some applications of graphs in different research domains.

- *Operations Research*: A large project consists of many smaller tasks with a precedence relation some tasks must be completed before certain others can begin. One graphical representation of such a project has a vertex for each task and an arc from u to v if task must be completed before v can begin.

- *Sociology and Sociobiology*: A business (or army, or society, or ant colony) has a hierarchical dominance structure. The nodes are the employees (soldiers, citizens, ants) and there is an arc from u to v if u dominates v. If the chain of command is unique, with a single leader, and if only arcs representing immediate authority are included, then the result is a rooted tree, as in Fig 3.2.

- *Genealogy*: A *family tree* is a digraph, where the orientation is traditionally given not by arrows but by the direction down for later generations. Despite the name,

a family tree is usually not a tree, since people commonly marry distant cousins, knowingly or unknowingly. However, it is always a DAG, because if there were a cycle, everyone on it would be older than everyone else on the cycle.

- *State Diagrams*: Let the vertices of D be a set of states of some process, and let the arcs represent possible transitions. For instance, the process might be a board game, where the states are the configurations and each arc represents the transition of a single move. Then walks through D represent *histories* that the process/game can follow. If the game can never return to a previous configuration (e.g., as in tic-tac-toe), the state diagram of the game is a DAG.

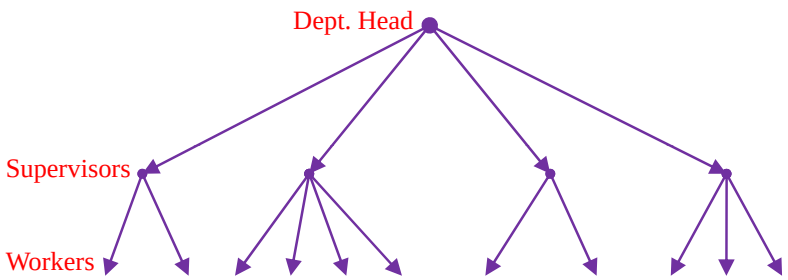

Figure 3.2: A corporate hierarchy

3.1.2 Graph algorithms

Graph theory algorithms can be traced back to the 19^{th} century, when Fleury gave a systematic method for tracing an *Eulerian graph* and G. The 20^{th} century saw algorithmic solutions to such problems as the minimum connector problem, the shortest and longest path problems, and the *Chinese postman problem*, as well as to a number of problems arising in operational research. In each of these problems we are given a network, or weighted graph, to each edge (and/or vertex) of which has been assigned a number, such as its length or the time taken to traverse it.

3.1.2.1 Related work

In the following we highlight some of the most important problems that researchers tried to solve by graph theoretic algorithms/approaches.

The Traveling Salesman Problem [22], in which a salesman wishes to make a cyclic tour of a number of cities in minimum time or distance, appeared in rudimentary form in 1831. It reappeared in mathematical circles in the early 1930s, at Princeton, and was later popularized at the RAND Corporation. This led to a fundamental paper of G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson [23] that included the solution of a traveling salesman problem with 49 cities. In the 1980s a problem with 532 cities was settled by Padberg and Rinaldi [24].

The Chinese postman problem, for finding the shortest route that covers each edge of a given weighted graph, was originated by Meigu Guan (Mei-Ku Kwan) [129] in 1960.

In matching and assignment problems one wishes to assign people as appropriately as possible to jobs for which they are qualified. This work developed from work of König and from a celebrated result on matching due to Philip Hall [130], later known as the *marriage theorem* [131]. These investigations led to the subject of polyhedral combinatorics and were combined with the newly emerging study of linear programming.

Graphs play an important role in the analysis of electrical networks. An electrical network is a collection of interconnected electrical elements such as resistors, capacitors, transistors etc. The behavior such as response to a unit impulse of an electrical network is a function of two factors [132]: 1) the characteristics of each electrical element and 2) how they are connected together i.e. their topology. It is the latter factor that brings graph theory into the picture. An electrical network is considered as a connected directed graph.

Most problems in analysis of a linear system are eventually reduced to solving a set of simultaneous, linear algebraic equations. These problems, usually solved by matrix methods, can also be solved via graph theory. The graph theoretic analysis of a linear system consists of two parts: 1) constructing a labeled, weighted digraph called signal flow graph, and 2) solving for the required dependent variable from the signal flow graph. A signal flow graph can be compared to a signal transmission network [132]. The signals travel along the edges and are multiplied by the weights of the edges traversed. The signal flow graph method of analysis is most useful when we want to solve for only one unknown variable.

Channel coding is an important building block in communication systems since it ensures the quality of service. Irregular repeat-accumulate (IRA) codes belong to the class of Low-Density Parity-Check (LDPC) codes and even outperform the recently introduced Turbo-Codes of current communication standards. IRA codes can be represented by a Tanner graph [133] with arbitrary connections between nodes of given degrees [134]. A Tanner graph is used to specify constraints or equations which specify error correcting codes. In coding theory, Tanner graphs are used to construct longer codes from smaller ones. Both encoders and decoders employ these graphs extensively [133].

Graphs play a major role in problems arising in the specification and translation of programming languages. A special kind of graph called a *finite automaton* is used in language theory to specify and recognize sets of strings called regular expressions. Regular expressions are used to specify the lexical structure of many programming language constructs. They are also widely used in many string-pattern-matching applications.

One of the most fundamental tasks in algorithms involving graphs is visiting the vertices and edges of a graph in a systematic order. Depth-first and breadth-first search are frequently used traversal techniques for both directed and undirected graphs.

The greedy algorithm for the minimum connector problem, in which one seeks a minimum-length spanning tree in a weighted graph, can be traced back to O. Boruvka [135] and was later rediscovered by J. B. Kruskal [136].

Graph algorithms were developed by G. B. Dantzig and D. R. Fulkerson [23] for finding the maximum flow of a commodity between two nodes in a capacitated network, and by R. E. Gomory and T. C. Hu [137] for determining maximum flows in multi-terminal networks.

Graph algorithms have been used to solve sequencing problems. For example [138], describes the sequencing problem in which n jobs with ordering restrictions have to be done by men of equal ability, by using graphs. This shows that how late only (but massive thenceforward) in the 20^{th} century the scientific interest in commodity flow problems arose.

Finding a longest path, or critical path, in an activity network dates from the 1940s and 1950s, with PERT (Program Evaluation and Review Technique) used by the US Navy for problems involving the building of submarines and CPM (Critical Path Method) developed by the *Du Pont de Nemours Company* to minimize the total cost of a project.

There are several efficient algorithms for finding the shortest path in a given network, of which the best known is due to E. W. Dijkstra [25].

By the late 1960s it became clear that some problems seemed to be more difficult than others, and Edmonds [26] discussed problems for which a polynomial-time algorithm exists. Cook [27], Karp [28], and others later developed the concept of NP-completeness. The assignment, transportation, and minimum spanning-tree problems are all in the polynomial-time class P, while the traveling salesman and Hamiltonian cycle problems are NP-hard. Further information can be found in [139].

3.1.2.2 Related work in SDR

In this section we discuss a tool called SynDEx (Synchronized Distributed Executives), developed by INRIA which is used in SDR design. This tool is based on graph theory. SynDEx entries for the design flow are two graphs:

1. a hardware platform diagram and
2. a software application diagram

In order to speed up the design phase, it is deliberately chosen to have only a restricted set of parameters and to take into account the architectural exploration:

- execution time of each IP
- atomic communication time for each data type
- communication media features

The goal is to obtain an implementation on the HW platform as quickly as possible, in order to deal with realistic constraints instead of having approximated simulations of those constraints in the design flow itself. Partitioning optimization of SynDEx is performed

using graph theory. The SW application graph is modeled by an extended Data Flow Graph (DFG), which is an oriented hypergraph. Each vertex corresponds to an algorithm operation or a processing element, which is activated by the fullness of its input buffer and each edge represents a data transfer between operations. An example of a SynDEx software application graph is given in Fig. 3.3.

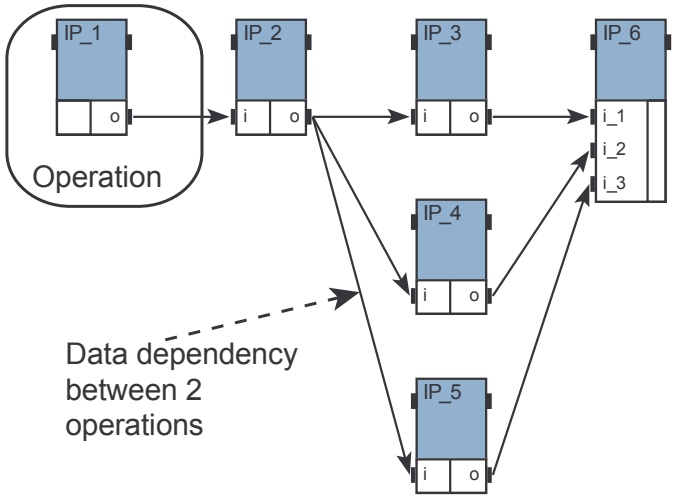

Figure 3.3: An example of SynDEx software application graph

Moreover, the model includes hierarchy, delay, conditioning (if/ then/ else) and factorization (for loops), in order to express the potential parallelism. Factorization, which is associated with a repetition factor, is used to repeat operations and requires additional specific vertexes in the DFG:

1. *Fork/Join vertexes*: the *fork* vertex explodes each element of the data it receives for each parallel repetition of the consumer operation, whereas the *Joint* vertex builds the data it sends, via the concatenation of each separated element produced by each producer operation repetition.

2. *Iterate vertex*: this vertex aims at sequentially duplicating a producer/consumer operation, where the outputs of the current operation can be the input of the next one, if the data names are similar. More details can be found in [30].

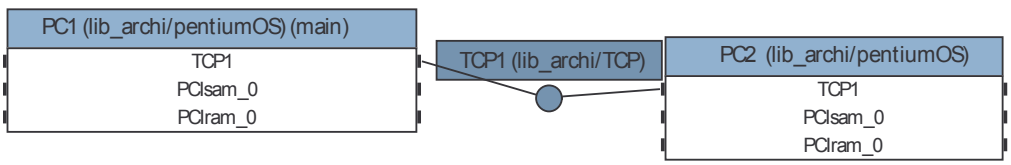

Figure 3.4: An example of SynDEx hardware platform graph

The hardware architecture is modeled by a non-oriented hyper-graph, where each vertex is a processor (hardware component) and each hyper-edge represents a communication medium, as shown in Fig. 3.4. In this model, a processor consists of one operator and as many communicators as there are connected media. An operator executes the operations, which are a part of the algorithm, and a communicator executes a communication operation, when a data transfer is required. By doing this, a multi-component architecture can be represented by a network of Finite State Machines (FSM) interconnected with communication media (FIFO, shared memories etc.).

The aim of this tool is to find the best combination of an algorithm which specifies the running of the application, and a multi-component architecture. In addition to this, real-time and embedding constraints must be satisfied. This methodology is based on a graph theory, in order to model the software application and the hardware architecture. The software and hardware are described by two distinct graphs. SynDEx transforms those two graphs with graph transformations in order to generate an optimized code implementation. An efficient implementation graph is obtained thanks to optimizations and simultaneous distribution and scheduling operations, while trying to minimize the execution time.

There are a large number of possible implementations. The optimization problem aims at selecting the most efficient one between all of them (best latency). The latency is the total DFG execution time on a given HW architecture, between the first scheduled operation and the last one. Moreover, the distribution and scheduling problem in the case of HW multi-component is known to be NP-hard (an exhaustive research on all the possible fulfillments is inconceivable). This is why heuristics are used to find the the optimal solution best approximation (greedy and genetic algorithm). The current heuristic approach attempts to minimize the total running algorithm execution time on the multi-component architecture. Moreover, since the Synchronized Distributed Executives (SynDEx) are automatically generated and safe, this consequently eliminates some of the tests and low-level hand-coding, and decrease the life cycle development as well. SynDEx provides a timing graph, as in Fig. 3.5, which includes simulation results of the distributed application and thus enables SynDEx to be used as a virtual prototyping tool.
SynDEx is also providing a static scheduling, which is of major importance for hard real-time requirements. It gives the guarantee of an execution within a restricted given latency period. SDR equipment will have very strong real-time constraints to respect. The introduction of SW in radio design must not cause customer's suspicion, or even worse, them rejecting it. This has already been witnessed in the mobile phone area with the collapse of WAP (Wireless Application Protocol), illustrating the fact that a badly introduced technological advance can turn into an economic disappointment.

Comparison between SynDEx and our approach

SynDEx uses a totally different approach than the one proposed in this thesis. SynDEx indeed addresses a mapping and scheduling problem on application graph to a hardware platform graph. Whereas we address a structural composition of the SDR design issue. To sum up, in SynDEx application graph, nodes are IPs and arrows are data exchanges, and in SynDEx hardware platform graph, nodes are processors and arrows are communication

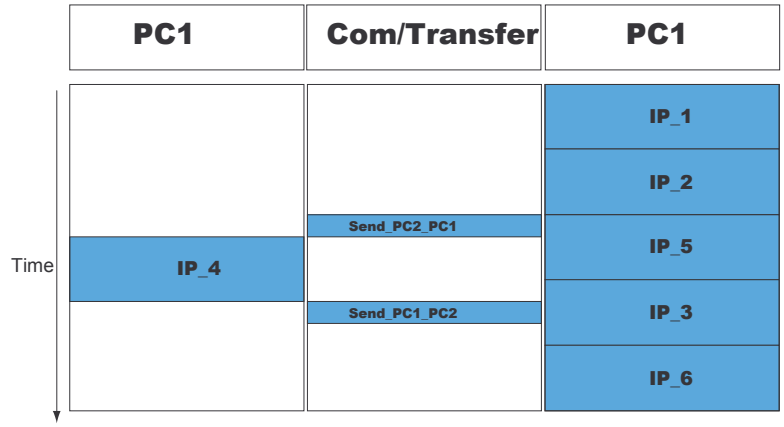

Figure 3.5: An example of SynDEx partitioning and scheduling

media. In the graph of this thesis, there is neither a notion of data exchange in arrows nor a hardware communication media representation. Arrows are just the expression of hierarchy: parent node may be made alone or by children nodes. It is just a structural view of the application (radio) at different levels of granularity.

Graph applications for partitioning and scheduling

Software partitioning is an important aspect in the design and operation of SDR. A fundamental graph partitioning algorithm was adapted and enhanced in [140], for applications in this domain. Real-time requirements have been derived from the radio interface timing for dedicated channels (DCH) of an IMT-2000W-CDMA transmission mode. The large variety of its potential SDR implementations has been modeled by a stochastic linear resource-runtime relation. Another application of graphs, particularly focusing on scheduling issues of SDR has been described in [141]. Authors describe the mathematical modeling of SDR design problems using directed acyclic graph (DAG), as the data flow can be represented in an abstract way by using a DAG. Further they analytically derive the speedup behavior of a DAG primitive that can be extended to bigger graphs. Both contributions [140, 141] are based on DAG, however, for applications in partitioning and scheduling.

3.1.2.3 Discussion

There are many computational problems about graphs, with important real-world applications, when the graphs have weights on their vertices and/or edges. For DAGs, many of these problems can be solved by essentially the same single-pass algorithm. This algorithm is the basic form of the sort of staged algorithm called dynamic programming in operations research circles.

In a DAG, the vertices can always be numbered consecutively so that all arcs go from lower to higher numbers. Using this numbering, many optimization problems can be solved by essentially the same algorithm, one that makes a single pass through the vertices in num-

bered order. For more general digraphs, algorithms for these optimization problems are less efficient or at least more complicated to describe.

Let us recall our aim. Firstly, our objective is to find the graph representation/model of multi-standards SDR systems; Secondly we want to optimize this graph for multi-standards and to do this we are to develop graphs with weights (costs) on their vertices.

Having discussed the basics of graph theory and highlighting some of the popular applications we come to the conclusion that the definitions of standard graphs already available in the literature cannot be applied directly to our problem. In order to graphically model the systems we are to incorporate some new modifications/dependencies that satisfy our needs and at the same time give us the opportunity to use the available graph algorithms.

3.2 Graph theoretical models of multi-standards SDR systems

3.2.1 Objective

In section 3.1.1, we described the basic terminology of graphs and in section 3.1.2, we explained some graph algorithms available in literature to solve some real world problems modeled as graphs. In this section we present our idea of developing graph theoretic models of multi-standards SDR systems.

We propose that a multi-standard SDR system can be represented as a graph with several layers. These layers depend upon the granularity level of considered processing elements (PEs). Our approach aims at providing the options to the designer, to select a set of operators, each of them at the most appropriate level of granularity as dictated by his needs. The expected advantage is that at certain levels of granularity, the operators can be re-used several times, inside and between different standards and the result will be an improved design of SDR system. We aim at helping the designer to make a granularity exploration for the design of multi-standards wireless equipments.

3.2.2 Definition of the graph structure

In order to achieve the objectives, described in above paragraph, we present here, our choice in terms of graph structure. Each node of the graph represents an elementary functional PE. In order to perform the processing of this PE, the designer may either directly implement the PE in the system, as an atomic piece (non divisible), or s/he can invoke lower-level smaller PEs. In this last case, several PEs may be necessary, or one PE called several times, in order to perform the processing corresponding to the higher-level PE.

In order to develop a graph model of multi-standards system, it is necessary to use a hypergraph instead of simple graph because in this hypergraph, we introduce two kind of dependencies between nodes, namely OR dependency and AND dependency as shown in Fig. 3.6 and Fig. 3.7 respectively. Node dependencies can occur between nodes of different levels. A node at a higher level (say at n), called a parent node, has dependencies

upon underlying nodes at lower level (say at $n-1$), called descendants/children nodes. An OR hyperarc (direct arrow) means that only one of the descendant nodes is necessary to implement the parent node. An AND hyperarc (inverted Y connection) means that all descendant nodes are needed to implement the parent node. It should be noted that there is no strict assignment of level and the only important information is the relative level between nodes that are interconnected, not their absolute level. In fact this notion of absolute level does not exist

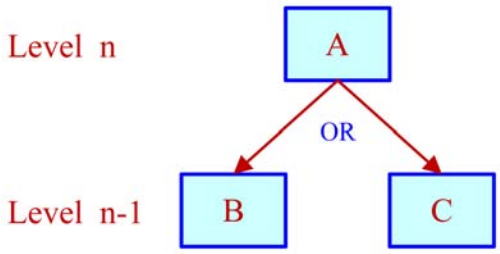

Figure 3.6: OR hyperarc

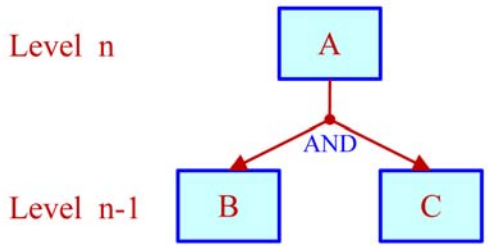

Figure 3.7: AND hyperarc

Using these hyperarcs we draw a generalized hypergraph corresponding to a conceivable tri-standard SDR system as shown in Fig. 3.8. The system is represented as a hypergraph of progressively simpler functional modules. Each node (module) represents a functionality that can either be implemented via a dedicated hardware/software component (e.g. an ASIC, or an intellectual property core), or can be achieved by invoking lower level modules. The hyperarcs leaving a node (parent) specify the simpler modules (descendants) that could provide the required functionality through multiple calls. Descendant nodes may not all be at the same level.

OR arcs from S1 show that S1 can be implemented through any one of A1, A2 or B1. An AND arc as pointing from S3 to A4 and A5, means that all descendant nodes are needed to implement the functionality of the parent node. In some cases, a parent node may have both AND and OR dependencies with its descendants. As a more concrete example, to realize an equipment that supports the standard S2, it is possible to realize the standard S2 in a unified block which is non divisible whether an ASIC or a program. When all the

equipment is built like that, then it is *Velcro* solution as we must completely switch to another block for all the calculations associated with other standard, in order to change the standard. Another possibility to implement the standard S2 is to implement nodes A2 and A3. We notice that now if the equipment is using standard S1, A2 may be retained and thus save a lot of effort in terms of reconfiguration compared to *Velcro* case, for example in terms of reconfiguration time. This is the primary interest of the approach proposed in this thesis.

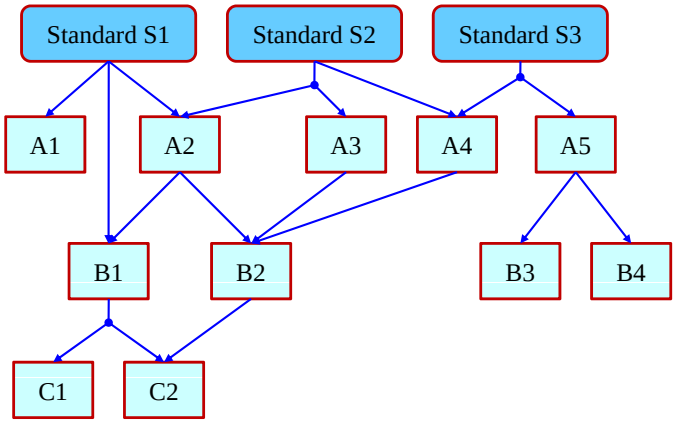

Figure 3.8: Generalized graph corresponding to a conceivable tri-standard SDR

3.2.3 Simplified example of tri-standard WiFi, WiMAX and UMTS transmitter

In this section we further illustrate above graph by selecting a design scenario in which we draw this graph for three different air interface standards. First selected standard is Wireless-Fidelity (WiFi). It is a logo from the WiFi Alliance that certifies network devices comply with the IEEE 802.11 wireless standards [142]. Wi-Fi/802.11 is very widely used (initially 802.11b [142], then 802.11g [143]), and now almost all laptops and other handheld devices came with WiFi built-in. Second chosen standard is Worldwide Interoperability for Microwave Access (WiMAX) [144]. It is an industry body formed to promote the IEEE 802.16 wireless broadband standard and provides certification for devices to comply with standard. Third selected standard is Universal Mobile Telecommunications System (UMTS) [145, 146]. It is a third-generation (3G) broadband, packet-based transmission of text, digitized voice, video, and multimedia. These standards are selected because of their worldwide use and popularity.

A global structure of the graph for these three standards, from transmitter side, is shown in Fig. 3.9. Graph in Fig. 3.9 is a simplified version of the complete graph with only a few nodes shown for illustration purposes. Of course the complete graph would be enormous showing all the nodes. At the top (coarsest grain level or highest level) of this graph are the communication standards themselves. Lower to this level are the building block

of these standards. At present, we have selected some of the important building blocks (transmitter side only) of the standards' physical layer (PHY) for clarity purposes, but ultimately, we aim at adding all the layers i.e., from application layer to PHY layer. As we move down to further lower levels in the graph, we pass through various granularity levels and finally we reach at the primitive elements level (finest grain level or lowest level). There might be many intermediate levels in between the top level and bottom level.

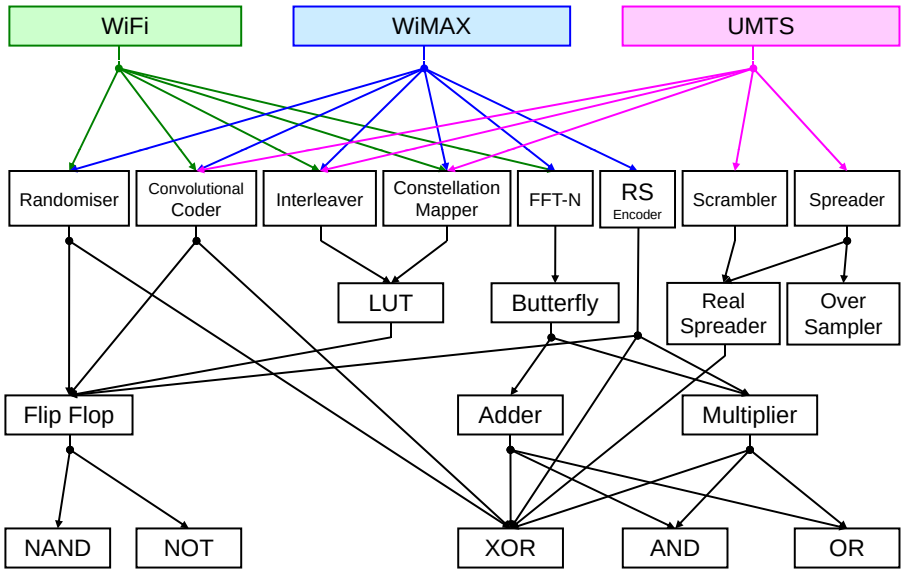

Figure 3.9: Global structure of the graph for tri-standard SDR system (transmitter side), simplified version

Let us explore the Fig. 3.9 in bit more detail. Each operation in Fig. 3.9 is processed sequentially for each new block of incoming data. For instance, to implement WiFi we are to use the common operators (COs). Let us assume that the selected COs are Flip Flop (FF), Butterfly and XOR. Procedure for selection of these operators is outlined in next chapters. We need to call a common operator different number of times to perform the functionality of WiFi. This is the way, how our graphical approach of Fig. 3.9 has to be interpreted. One box e.g. Butterfly, corresponds to one call. For instance, to implement FFT-N, this box will be called $(N/2) \times log_2 N$ times. $(N/2) \times log_2 N$ is a parameter attached with the arrow from FFT-N to Butterfly block just as different parameters are associated with arrows as explained in next chapters. Inside each operation there is a need to make scheduling as it is necessary to call FF/XOR/Butterfly operators several times. This scheduling issue which is a consequence of common operators approach has been partially addressed in [118].

From the graph of Fig. 3.9, it is clear that there are many ways to perform the functionality of a PE and to draw all these possibilities in a graph is a tedious job. This is not the purpose of this thesis to develop complete graphs. Graphs that will be used are provided by other studies e.g. identification of joint operations, which is also a task in itself, inde-

pendent of the work of this thesis. Due to this fact, we often consider sub-graphs of the total graph that does not present standards at the top of the graph, and not necessarily come to arithmetic operators and bottom of the graph. Let us consider a more specific scenario of a complete hypergraph of transceiver for multi-standards SDR system. In this scenario, we restrict ourselves to three main communication tasks i.e. channelization, equalization and OFDM and draw a partial hypergraph. These communication tasks are required by most of the air interfaces. Various alternatives to do perform these functions are shown in Fig. 3.10.

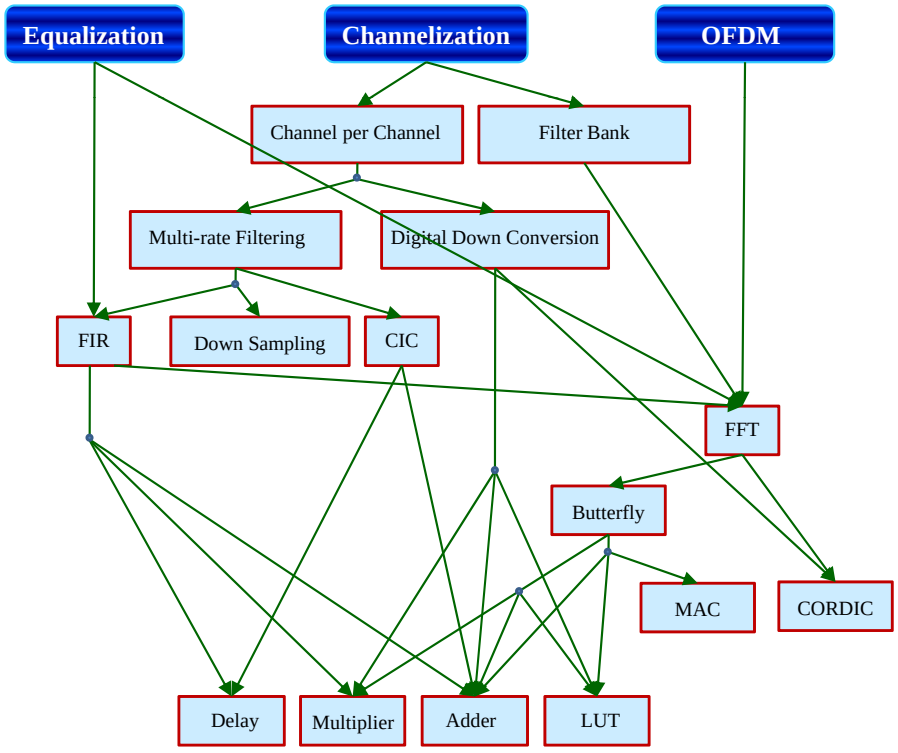

Figure 3.10: An example of partial hypergraph

Looking at this graph gives us the pictorial view of numerous alternatives that are available to do the same task. This illustrates that the number of design alternatives can grow very quickly. Although, we can estimate or intuitively deduce optimal choices on small graphs, it becomes very difficult for bigger graphs. For instance, for channelization, we can use either channel per channel or filter bank approach. If we choose the filter bank approach then this can be done by FFT. The same FFT can be used by OFDM and equalization as well. Now the questions that comes in our mind are; *We have different alternative but which one is the best*? *Best according to which criteria? Which path in the graph will give us the optimal solution*? To do this, first we have to put weights i.e. costs on the nodes of our graph in order to derive a cost function and then we have to run some optimization algorithm to find the optimized path. These cost parameters and costs are discussed in chapter 4 while optimization algorithms are discussed in chapter 5.

3.3 Network theory reformulation

3.3.1 Motivation

In the present section, we explore the possibilities of finding some smart algorithms which fully utilizes the special structure of the graph to efficiently search the solution, by recasting our problem as a single- source (or single terminal (sink)) network-design problem. This move of ours' is motivated by the fact that the network-design problem is well known and counts with a very extensive literature, which offers algorithms, computational complexity results and many practical applications in a variety of contexts such as telecommunication, transportation, water resources and even social interactions [147].

Below, we show how to perform the problem reformulation and give a specific example grounded on a practical design. In support of our approach, we identify and discuss several specific algorithms which seem particularly well suited to our purposes [148, 149, 150].

3.3.2 Network design problem

A simple illustration of the single-source network-design problem is shown in Fig. 3.11. Suppose that there is an initial point called the origin, O, and three terminals (destinations), t_1, t_2 and t_3. We wish to connect O to each t_i in a way that satisfies certain optimality criteria. There may also be some intermediate nodes, which themselves may be (partially) interconnected. There are many conceivable solutions. The most obvious one is simply to build three direct roads, one from O to t_1, a different one from O to t_2 and a third one from O to t_3. This solution is certainly plausible and probably provides the shortest travel times, but it is unlikely to be ideal. By building dedicated roads, we are possibly missing the savings resulting from having common segments which may be used by all traffic, regardless of the destination. For example we could build a road from O to some intermediate point A, plus dedicated segments from A to each t_i. Because the segment $O \rightarrow A$ is used by all traffic, certain savings may result (although this may lengthen the travel time to all traffic).

By the same reasoning, building dedicated segments from A to each t_i may not be optimal. Perhaps we can build a dedicated road $A \rightarrow t_3$, but also a segment from A to some intermediate point B, followed by segments $B \rightarrow t_1$ and $B \rightarrow t_2$. In the latter proposal, all traffic would travel over the $O \rightarrow A$ segment, the t_3 traffic would go directly over $A \rightarrow t_3$ while both the traffic to t_1 and to t_2 would continue over $A \rightarrow B$, and from B over $B \rightarrow t_1$ and $B \rightarrow t_2$, respectively. There may be many other possibilities

3.3.3 Graph to network conversion

Our problem can be recast as a network-design problem by proceeding as follows. The communication standards correspond to terminals we wish to reach. Building a direct road from the origin to each terminal corresponds to the *Velcro* solution: choosing a self-contained dedicated complex component to support each standard. This will provide the *fastest routes* to be sure, but probably not the ideal solution. Installing a self-contained component of *medium* complexity is the equivalent of building a road from the origin to the intermediate point A in the example above. For instance, [17] shows that many important tasks of a communication receiver can be implemented through the FFT as mentioned

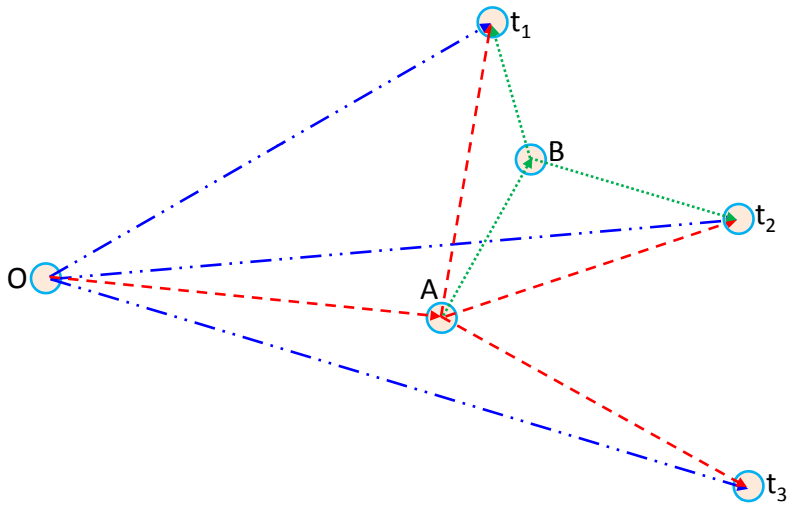

Figure 3.11: A simple single-source network design problem

earlier. In turn, the FFT functionality can be implemented with a butterfly operator. Thus, we can *build a road* from the origin to the FFT (i.e. install a self-contained FFT component) to be shared by several tasks. This will likely reduce the overall cost of the design, although it will generally *increase the travel time*. But we could also *reach* the FFT node (i.e. provide the functionality of the FFT operator), by first *building a road* from the origin to the intermediate junction *butterfly* and from there utilize an *existing road* to the FFT (i.e. install a self-contained butterfly component, and invoke it repeatedly to provide the FFT functionality).

The network-design problem, corresponding to a section of the partial hypergraph of design choices given by Fig. 3.10 is shown in Fig. 3.12. The parts of the graph that concern lowest level components and the channelizer have been ignored for pedagogical reasons.

Recall that now the objective is to build a *road network* that provides a path from the origin to each *terminal*. The solid arrows mark *free* (pre-built) roads that take time to travel, but require no monetary expenditure. These represent known algorithms which can be used to provide a higher level functionality. For example from the FFT node there is a pre-built (free) *road* to OFDM, another to equalization and yet another to the FIR node, because there are known algorithms that allow the implementation of equalization, OFDM and FIR through the FFT operator [17]. The dashed arrows indicate *possible roads* that definitely cost money to build (but possibly negligible time to travel).

Building a road means installing a self-contained component to provide the functionality pointed by the arrow. For example there is a dashed arrow from the origin to FFT (meaning we can provide the FFT functionality by installing a self-contained FFT component). There is also a *possible road* from the origin to butterfly. If we build this road (i.e. if we install a self-contained butterfly component) we can reach the FFT node via a free (algorithmic) path (shown by a solid arrow from butterfly to FFT), because one can provide the FFT functionality by repeatedly invoking a butterfly operator. Fig. 3.13 shows in

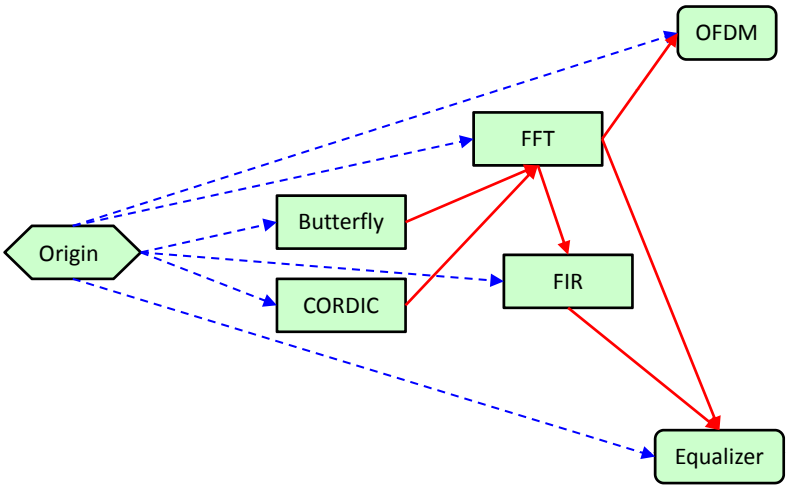

Figure 3.12: The network-design version of section of partial hypergraph

greater detail the *network-design problem*, corresponding to Fig. 3.10.

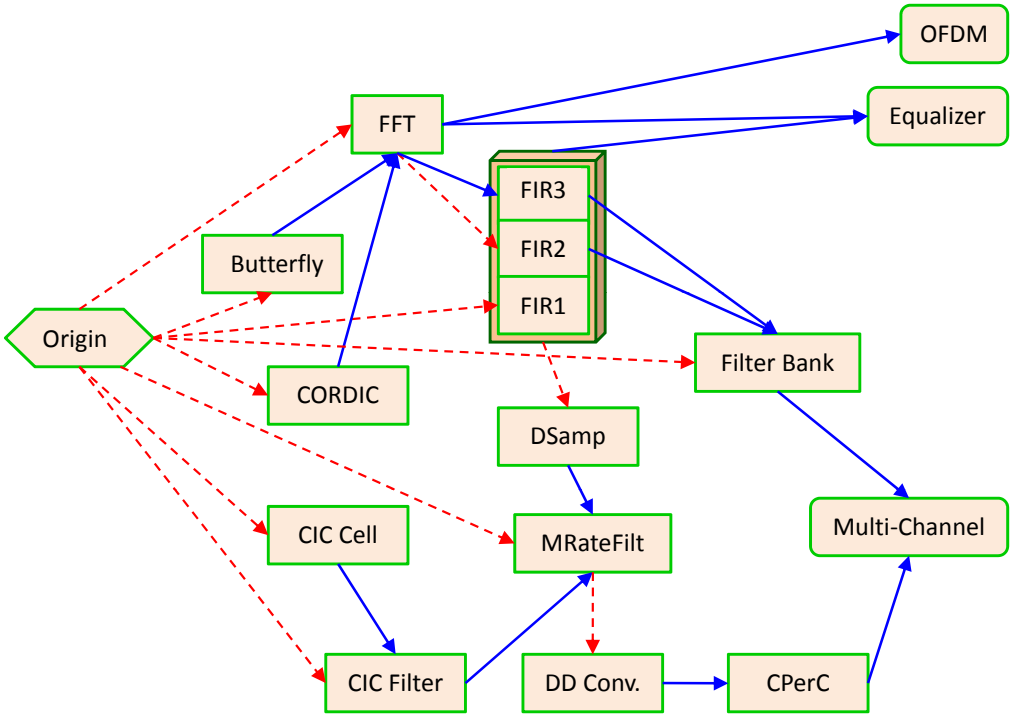

Figure 3.13: A network view of partial hypergraph given in 3.10

The fact that the digital down conversion module can be done through CORDIC, and the lowest level modules are not considered, for clarity. And some obvious dashed arrows from the origin (e.g. from the origin to equalizer, meaning installing an equalizer component)

are omitted to reduce clutter. The main purpose of Fig. 3.13 is to show that representing the AND dependencies is challenging in some cases, and may require some ingenuity and additional research. The FIR *triple node* illustrates some of the main issues. A *triple* FIR node is used in Fig. 3.13 to show that there are three different *routes* to get the FIR functionality, and the route has an impact on what can be done from there. The *sub-node* FIR3 concerns the case where the FFT is used to implement the FIR. The *sub-node* FIR2 refers to the case when an FIR component is installed, even though the FFT functionality is present (this may be done to reduce execution time). FIR1 denotes a case in which an FIR component is installed and the FFT functionality is not available (perhaps OFDM and equalizer dedicated components have been installed, and the FFT module is unnecessary). In the cases FIR2 and FIR3, the functionality of the filter-bank module can be achieved algorithmically, because both the FFT and the FIR are available. However, under FIR1, there is no FFT module, thus, the filter-bank functionality would require an installed component. For this reason, there is no arrow pointing to the filter-bank module from the FIR1 sub-node. On the other hand, the solid arrow pointing to the equalizer from the FIR is drawn from the oval, around the various FIR sub-nodes. This indicates that we can achieve equalization through the FIR, irrespective of how we achieve the FIR functionality (contrary to the filter-bank case).
A simpler case of AND dependency is also shown in Fig. 3.13, which involves achieving multi-rate filtering (MRateFilt) through both FIR and down sampling (DSamp). The dash line from the FIR oval to DSamp, followed by a solid arrow from DSamp to MRate-Filt, shows that if we have the FIR functionality (no matter how) and we install (build a road to) a down sampler, then we can achieve multi-rate filtering *for free* (algorithmically).

The reformulation of this provides us with several well-studied algorithms. A network-design problem with two metrics: cost and distance is described in [148]. This *fits* well with our formulation, with execution time as *distance* but this algorithm is probabilistic. [149] modifies it to make it deterministic, utilizing linear programming. The basic idea of [148] is as follows: The algorithm has several stages. At each stage, it forms a matching (pairing) of terminal nodes (not previously eliminated). Then, of each pair of nodes, it chooses a *center*. At this point, the non-center nodes are removed from further consideration and a new stage of the algorithm starts (with a new matching among the centers only). Because for us, the terminals correspond to standards to be supported by the radio, they should be relatively few. Thus, this algorithm will give an answer in a relatively short number of stages. The algorithm provided in [150] minimizes costs, while keeping length (time/latency) under a specified budget. These and many other algorithms available in the rich network-design literature are to be explored to find the best fit for our purposes. However because of the problems involved in the conversion of the hypergraph to network-design especially from the AND hyperarc point of view because it fast becomes impossible to manage complex AND hyperarc relations as mentioned earlier, we keep our energies focused on the hypergraph design for multi-standards systems.

3.4 Conclusions

After stating the common operators' technique of parametrisation and then illustrating the theoretical approach of finding common operators in chapter 2 we tried in this chapter

to address the question of defining a suitable graph model of multi-standard SDR systems. After exploring different possibilities we came up with the idea of modeling the graph as hypergraph to meet our needs. We also explored an other perspective of conceiving the hypergraph as a network design problem which was a study done by a post doctoral fellow in our laboratory [20]. Due to the unsuitability of the network design approach because of complex relations of AND hyperarcs we reverted back to the hypegraph approach to model multi-standards SDR systems. After having the graph, now we are faced with the task of putting the weights to the nodes and arcs of hypergraph. Theses weight parameters that are to be associated with graph entities i.e. with nodes and arcs are discussed in detail in chapter 4.

Chapter 4

Cost Parameters and Costs for Optimization of Multi-Standards SDR Systems

Contents

4.1 Introduction

Graph modeling of different SDR systems gives us a pictorial view of available alternatives to perform the same tasks. To determine that which alternative is best suited for a given task we try to convert this graph in an optimization problem. The graphs can be translated in a boolean formula with respect to AND and OR arcs and then can be solved by some optimization technique. To do this we are to find and associate the necessary cost parameters and costs with graph models.

In this chapter we start our discussion with different cost parameters and types of the costs. These costs can be analytical costs or implementation costs. Analytical costs depend upon the mathematical equations/operations behind a particular task while implementation costs depend upon the chosen target technology. Then we discuss the digital

hardware composition of SDR in section 4.3.2.1 which is a key design step in its creation. The hardware design is, of course, much more complex for a software radio than a conventional radio because of the software radio's additional capability. In section 4.3.2.4, we provide a brief summary of general algorithmic partitioning for three foundational digital hardware available: DSPs, FPGAs and ASICs. We describe which tasks are suitable for which hardware technology. Each exhibits a certain level of re-programmability, that is, an ability to alter device hardware or software. Based on the selected costs associated with different graph components we formulate our cost/objective function which is presented in section 4.4. Our objective function represents a multi-objective optimization problem (MOP). Objective functions corresponding to MOPs form a mathematical description of performance criteria which are usually in conflict with each other. Finally a conclusions' section ends this chapter.

4.2 Cost parameters

There can be many cost parameters, among them, we consider that the *building cost* (BC) of processing elements (PE), *computational cost* (CC) of the PE of the systems and *Number of Calls* (NoC) are the principle ones that are to be associated with entities (PE and arrows) of graph models.

The two identified cost parameters that we think appropriate to be associated with *processing elements* (PE) of the graph are explained below:

- Building cost of processing element that is capable of computing a function. By building cost of the PE we mean its installation cost. This cost can be considered as implementation cost of PE. BC is paid once during the useful life of a radio system. If it is required to call several a PE in equipment, the BC associated with element is not multiplied. Re-use of PEs thus provide an idea of cost reduction.
- Computational cost of the PE. By computational cost of the PE we mean the time taken by a PE to compute a function. This cost is incurred every time a PE is invoked or called by higher PE.

These costs can be thought of in terms of number of multiplications, number of additions, number of LUT, number of slices, number of gates, number of execution cycles, area, power, execution time, just to name a few.

Let us consider the cost parameters that are to be associated with arrows of the graph. The graph of any system consists of various granularity levels [118]. The task at higher granularity level can be performed with the aid of components that are at lower granularity level with less building cost but with more time. Hence we call the parameter associated with the arrows, *number of calls (NoC)*, i.e. number of times a processing element at lower granularity level $n-1$ will be called to perform the task of the component at higher granularity level n. This means that same task can be performed by a PE at level n instead of PE at level *n-1*, in less time but with much higher cost and vice versa.

In general, a node having higher hierarchy has a higher building/manufacturing cost than the nodes with lower hierarchy that it calls. Similarly, the execution time of a node with higher hierarchy is usually lower than all the executions and calls needed to be done to make the same calculation using the lower hierarchy nodes.

To elaborate further, we consider the concrete case of a hardware implementation, where BC represents a cost in number of gates (or area), as for an FPGA or digital ASIC implementation. In the example of a filter processing, there are many design options. Some designers will try to parallelize their architectures in order to gain in speed of execution but at the cost of an additional area. This is the case of the higher hierarchy element. However, in order to save in terms of area (building cost), it is possible to call several times a single unit of type MAC (Multiply Accumulate), which would be the element of lower hierarchy with a much lower area cost (building cost) and may also have a lower computational cost, but which will be required many times (number of calls from the arc connecting the filter to the MAC) in a sequential manner to perform filtering. The resulting computational cost becomes more expensive than the building cost of the filter alone. Hence it is clearly evident that we need to find compromises; 1) between complexity and speed of execution and 2) between the top and bottom of the graph i.e. between fine and coarse granularity.

For a deeper insight, let us explain Fig. 4.1, that reconsiders the example of Fig. 3.10 with parameters' values that are chosen arbitrarily. Fig. 4.1 shows the graph model of a standard that requires OFDM, Equalization and Channelization. Costs are associated with entities of the graph of Fig. 4.1. PEs are tagged with BC/CC e.g. Multiplier is tagged with 10/5 i.e. building cost is 10 and CC is 5. Arrows are tagged with NoC e.g. x10 means that higher PE will call the lower PE 10 times. It is to be noted that these parameters' values are not the real ones but chosen arbitrarily. They have no reality here in terms of time or number of gates, but they follow a certain logic on relationships between parameters of the same nature. We see that an FIR can be used for a building cost of 500 and a computational cost of 1000, or an alternative design is to implement the three operators Delay, Multiplier and Adder for a total building cost of $1 + 10 + 4 = 15$ and a computational cost of $1 + 5 + 2 = 8$ (if we consider a sequential execution because of data dependencies). But it is necessary to call each of three operators 180 times to perform the functionality of FIR with a total execution cost of $180 \times 8 = 1440$.

In some SDR architectures, BC can also be represented by the number of logic units required by an FPGA implementation e.g. number of ALUTs etc. This allows several standards to use the same components and share their costs and this is exactly what the concept of common operators is all about. The CC parameter requires more consideration. Some nodes will be implemented by invoking several descendant nodes, each a certain number of times, where we may use an average number if this is not constant.

4.3 Types of costs

In this section we investigate a bit further the available choices for different building costs. Determining these costs is a very importance step in optimization procedure, and hence,

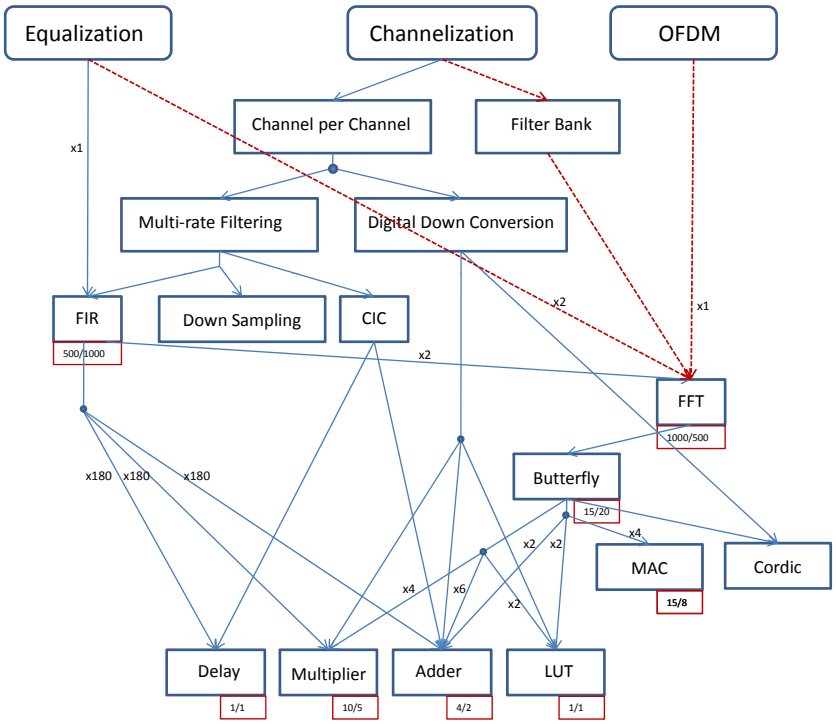

Figure 4.1: Graph model with associated costs; PEs tagged with BC/CC and arrows tagged with *NoC*

a fair number of various possibilities have been considered. The costs can be divided into three major categories.

- Analytical costs
- Implementation costs
- Execution costs

These costs are discussed in detail in the following section.

4.3.1 Analytical costs

These costs are a result of the analysis of computational complexity of different PEs composing air interface standards. This analysis gives us the estimates of number of required basic operations per output data item in each function. The basic operations include the following:

- Arithmetic operations
- Logic operations and
- Memory load/store operations

Analytical costs are based on actual calculation of above mentioned basic operations. These costs can be quite different from implementation costs. The implementation costs will be based on selected hardware for SDR implementation as discussed in section 4.3.2.

Complexity evaluation of IEEE 802.11a and UMTS

In this section we present the computational complexity of major transceiver tasks of IEEE 802.11a [151] by estimating the number of required basic operations per output data item in each task. We also present the computational complexity of UMTS (FDD mode), Uplink transmitter blocks and Downlink receiver blocks, according to particular test cases defined in 3GPP specification documents [145, 146]. The analytical costs of different transmit and receive tasks of IEEE 802.11a (by evaluating the computational complexity) are given in Table B.1 and Table B.2 respectively of Appendix B. The complexity figures mentioned in Tables B.1 and B.2 are computed with the help of [152, 153]. The results of analysis in case of UMTS, are given in Appendix B's Table B.3 and Table B.4. The complexity figures mentioned in Table B.3 and Table B.4 are derived from [154]. Having the computational complexity at hand now, here comes one major question; *How we can use these complexity figures as costs?*

As far as the complexity figures are concerned, we have different types of basic operations for different functions in selected standards. From the computational analysis of IEEE 802.11a presented in Table B.1 and Table B.2 it becomes clear that on the transmitter side IFFT is the most computationally complex task while Viterbi decoding and FFT dominates the complexity on the receiver side. Based on this conclusion we decided that number of multipliers and adders can be used as BC. The tasks that are related to LUT accesses e.g. Interleaving etc., can be considered to be some sort of addressing scheme and is not computationally intensive. In case of UMTS also, we have counted only the arithmetic operations. Logical operations have been neglected because they are basis cells in hardware and thus much simpler than multiplier or adder. The complexity figures shown in Appendix B's Table B.1 and Table B.2 can be drawn for IEEE 802.16 with minor difference in some functions e.g. FFT etc.

4.3.2 Implementation costs

We can envision that the implementation costs are a direct consequence of selected hardware for the implementation of SDR terminals. Details of these available hardware with their pros and cons are presented in the following section.

4.3.2.1 Digital hardware choices

Requirements for SDR's signal processing are very demanding. Tasks for SDR are quite heterogeneous, ranging from data flow processing of the baseband signals, error correction on bit level, sophisticated protocol handling, and on top level, the operating systems for applications. This implies that several processor technologies should be used for SDR designs. For maximum flexibility, all functions of SDR mobile terminal should be defined in software. However, because of computing requirements, implementing them in software

over strains the possibilities of even sophisticated VLIW DSPs. Furthermore, power consumption, silicon area, and last but not the least, engineering time and development costs for new products must be minimized.

Real-time SDR hardware can be built using a variety of digital hardware consisting of FPGAs, DSPs and ASICs. While a DSP represents the most generalized type of hardware that can be programmed to perform various functions, an ASIC is the most specialized piece of hardware and can be used only in the specific application for which it has been designed. An FPGA offers a compromise in flexibility between an ASIC and a DSP.

4.3.2.2 Field programmable gate arrays

The first breakthrough in SDR came with the use of FPGAs. FPGAs were developed in the mid-1980s as an alternative to glue logic arrays and DSPs. FPGAs are commonly used in logic emulation and prototyping in most systems. They are ideal for prototyping since the chips are reusable after the bugs have been fixed or the system has been upgraded. Though software simulations can verify the accuracy of a circuit, they do so at a much slower rate than logic emulation circuits using FPGAs. Though the mapping of the circuit to the FPGA takes a long time to complete, once it is done, the system can evaluate the circuits at millions of cycles per second.

In the case of software radios, FPGAs have the additional advantage of adding adaptability and flexibility in the final product. FPGAs at times can also help conserve silicon area since one chip can be configured to perform more than one function and the configurations can be changed on-the-fly. The following is a list of situations in which the use of FPGAs in digital signal processing applications is most beneficial:

- Systems with high sample rates
- Systems involving variable or non-conventional word length FPGA-based DSP designs because they are more efficient since the word length on the FPGA can be set exactly to the required length
- Systems with very-high-order FIR filters because the algorithm can be implemented in parallel decreasing the time required
- Systems with fast correlators because the LUT architecture of FPGAs provides a fast and efficient way to build correlators.

In the case of FPGAs the cost parameters that we can use are number of slices, number of CLB, number of LUT, number of Logic Cells, number of Flip Flops, etc. In design scenario presented in section 6.4 we use number of ALUTs as building cost of the communication block (CB)/processing element (PE). The design scenario presented in section 7.4.2 uses number of logic cells (LC) as building cost of CB/PE. In these scenarios we have considered the building costs that are based on the FPGA implementations. We may consider these costs based on DSP implementations (number of cycles). However, we prefer to use the costs based on FPGA hardware implementation. The reasons for doing this are explained in the forthcoming section 4.3.2.4.

Table 4.1: IEEE 802.11a and IEEE 802.16 implementation on FPGA

Area Metrics for a XC2V3000-4FG676 Device				
	IEEE 802.11a		**IEEE 802.16-2004**	
Parameter	Used	%	Used	%
Number of Slices	1678	11	2614	18
Number of Slice Flip Flops	2353	8	3566	12
Number of 4 input LUTs	2814	9	4304	15
Number of Bonded IOBs	29	5	29	5
Number of BRAMS	12	12	12	12
Number of GCLKs	1	6	1	6

Example of FPGA based costs

In this section we present an example of FPGA based costs as discussed above. In Table 4.1 we have costs in terms of different parameters of FPGA e.g.number of slices, number of LUTs, etc. These parameters can be used as building costs in our optimization tool. In this table costs are given for the full implementation of IEEE 802.11a and IEEE 802.16-2004 on selected device of Xilinx family of FPGAs [33, 34]. However, we can get these cost figures for different functions of these air interface standards e.g. scrambling, convolutional coding, interleaving etc. This point is further illustrated in part III of this thesis where design scenarios are discussed.

4.3.2.3 Digital signal processors

The DSP is essentially a microprocessor optimized for digital signal processing applications. DSPs are flexible because they can be programmed repeatedly even with a high-level language (HLL) like C. Modifications and upgrades can be made via this HLL, reducing the design time for each iteration. The flexibility, however, sometimes comes at the cost of efficiency. For instance, applications requiring several computations that could be done in parallel may have to be broken into sets of sequential computations based on the number of multipliers/accumulators available on the DSP.

Even if the DSP has provisions for parallel execution of instructions, like parallel multiply and accumulate (MAC) DSPs, the size and number of units are fixed, and they may not be optimal for a particular task. In general, the more processing per data sample, the more cycles needed and the slower the data is processed. One way to reduce the execution

Table 4.2: DSP Blackfin processor performance on DSP functions

Function	**Cycle Count**
N sample T tap real FIR filter	NT/2
N sample B stage Biquad IIR filter	5NB/2
N sample T tap convolution	NT/2
N dimensional vector sum	N
Complex FFT Radix-2 butterfly	3
256-point complex FFT (Radix-2)	3176
256-point complex FFT (Radix-4)	2630
Viterbi for GSM (16 states, 378 soft decision symbols)	6069

time is to employ more than one DSP to implement an algorithm. However, coordinating these DSPs is a challenging task for the programmer. Other solutions include using less generalized hardware, which can be configured to optimize a particular application.

Gate arrays offer the designer a higher degree of parallelism than DSP. An ASIC goes a step further than the gate array, where the entire IC including the positioning of the logic gates are determined by the designer. ASICs are generally used when a DSP has insufficient processing power or when the system has a sufficiently high volume of production to justify a custom solution.

Example of DSP based costs

In this section we present an example of DSP based costs as discussed above. In Table 4.2 we have costs in terms of cycle counts for a Blackfin processor. The Blackfin DSP processor [155] is a recent technology development targeted at cellular handset applications and is very appealing for SDR development [62]. Number of cycle counts can be used as execution costs in our optimization tool. However finding the costs in terms of cycle counts is more difficult than finding the costs in terms of FPGA parameters. Determining the BC in case of DSP based solution is extremely difficult. For the BC, when system implementation on DSP is considered, the designer may put a fixed price cost for the hardware architecture of the DSP.

4.3.2.4 Algorithmic partitioning of typical transceiver tasks

ASICs to FPGAs to DSPs represent the options for a designer ranging from high speed and minimum flexibility to maximum flexibility and very little hardware optimization. While general communication problems of relatively low complexity can be solved with a DSP, FPGAs and ASICs tend to be more useful when the complexity of the problem increases as in 3G and 4G wireless communication systems or when the product has to be manufactured in large volumes.

In this section we explore form algorithmic point of view that why designer of SDR systems consider FPGAs as their first choice? A multi-standard SDR terminal require the implementation of several protocols as desired by designer. Wireless protocol implementations tasks can be divided into two algorithmic categories:

- Control Flow
- Data Flow

The data flow part is delineated by extensive arithmetic computations on data streams. These algorithms provide most of the processing needs of the standards. These algorithms provide implicit parallelism and are best suited to run on hardware that can exploit this parallelism such as reconfigurable hardware. The tasks for several standards with the best hardware technology to realize them, are given in Table. 4.3.

We are concerned here with the data flow category since our work is related to the optimization of air interface protocols at physical layer level. Apart from other air interface we mainly concentrate on the systems based on OFDM because of the advantages that we may have as explained in design scenarios of part III. Hence in design scenarios, cost parameter of PE are principally based on reconfigurable hardware/processors i.e. FPGAs although other options can also be explored. This choice is based on the fact that the functions that are related to the data flow are considered to be performed best by FPGAs. Arguments of Table. 4.3 [62] reinforces our choice.

4.3.3 Execution Costs

Execution cost is clearly a function of time required by a processing element to do its own calculations in order to achieve its functionality. The execution time can therefore act as a good execution cost, whatever is the target of implementation. In the case of implementation on a processor, the number of cycles necessary for the execution of a processing element can also be considered a cost execution.

4.4 Approaches to formulate cost/objective function

Multiobjective optimization also known as multi-criteria or multi-attribute optimization is with no doubt a very important research topic both for scientists and engineers because of the multiobjective nature of most real world problems. In Operations Research more than 20 techniques have been developed over the years to try to deal with functions that have multiple objectives, and many approaches have been suggested, going all the way from

Table 4.3: Algorithm partitioning of typical baseband processing tasks

Technology	**Tasks**	**Arguments**
DSP/ Microcontroller	Protocol stacks and management. Portion of channel estimation, control flow type of processing.	Complex software structure with large decision trees. Requires little raw arithmetic or not continuous usage. Signal processing tasks that do not require continuous processing can be executed.
Hardwired functions	FEC CDMA Analog RF	Viterbi decoder or turbo decoder Hardwired functions require minimum energy and area Currently no programmable solution expected.
Reconfigurable Architecture	Rake receiver, correlations, searcher, header detection, channel correction, downsampling, FFT, demodulator, short PN code	These algorithms require most of arithmetic processing power. Requires real time processing of data streams

naïve combinations of objectives into a single one to the use of game theory to coordinate the relative importance of each objective. However, the fuzziness of this area lies on the fact that there is no accepted definition of *optimum* as in single-objective optimization, and therefore is difficult to even compare results of one method to another, because normally the decision about what the *best* answer corresponds to the so-called *human decision maker*.

The approach of combining objectives into a single function is normally denominated aggregating functions, and it has been attempted several times in literature with relative success in problems in which the behavior of the objective functions is more or less known [35, 156]. There are many popular aggregating approaches like *weighted sum approach*, *goal programming*, *the ϵ-constraint method*, etc.

Weighted sum approach was the first technique for the generation of non-inferior solutions for MOPs. This method is very efficient (computationally speaking). The problem with this approach is how to determine the appropriate weights when we do not have enough information about the problem. Goal programing may be a very efficient approach if we know the desired goals that we wish to achieve. However, the decision maker is given the task of devising the appropriate weights or priorities for the objectives that will eliminate the non commensurable characteristics of the problem, which is most cases is difficult unless there is prior knowledge about the shape of the search space [35, 156]. ϵ-constraint method is based on minimization of one (the most preferred or primary) objective function, and considering the other objectives as constraints bound by some allowable levels ϵ_i. This method is time consuming, and coding of the objective functions is difficult, or even impossible for certain problems, particularly if there are too many objectives [35, 156].

Hence we consider only the weighted sum approach to define our objective function because of its popularity and simplicity.

4.4.1 Weighted sum approach

This method consists of adding all the objective functions together using different weighting coefficients for each of them. This means that our multiobjective optimization problem is transformed into a scalar optimization problem of the form:

$$\min \sum_{i=1}^{k} \omega_i f_i(\bar{x}) \tag{4.1}$$

where $\omega_i \geq 0$ are the weighting coefficients representing the relative importance of the objectives and f_i represents the i^{th} objective function of vector of decision variables $\bar{x}$. It is usually assumed that,

$$\sum_{i=1}^{k} \omega_i = 1 \tag{4.2}$$

Since the results of solving an optimization model using equation (4.1) can vary significantly as the weighting coefficients change, and since very little is known about how to choose these coefficients, a necessary approach is to solve the same problem for many different values of ω_i. But in this case, the designer is still, of course, confronted with the decision of having to choose the most appropriate solution based on his/her intuition.

Note that the weighting coefficients do not reflect proportionally the relative importance of objectives, but are only factors which, when varied, locate points in the Pareto set [157]. For the numerical methods that can be used to seek the minimum of equation (4.1), this location depends not only on ω_i values, but also on the units in which the functions are expressed.

If we want ω_i to reflect closely the importance of all the objectives, all functions should be expressed in units of approximately the same numerical values. Additionally, we can also transform equation (4.1) to the form:

$$\min \sum_{i=1}^{k} \omega_i f_i(\bar{x}) c_i \tag{4.3}$$

where c_i are constant multipliers that will scale properly the objectives.

Many applications that uses this technique can be found in literature. Syswerda and Palmucci [158] used weights in their fitness function to add or subtract values during the schedule evaluation of a resource scheduler. Wilson and Macleod [159] used this approach as one of the methods incorporated into a GA to design multiplier-less IIR filters in which the two conflicting objectives were to minimize the response error and the implementation cost of the filter. Jakob *et al.* [160] used a weighted sum of the several objectives involved in a task planning problem. Some other examples can be found in [161, 162, 163].

Based on the approach described above we develop our objective function in following section, and this function will be optimized by our optimizer tool.

4.4.2 Objective function

In this section we describe the development of our objective function. There are two key parameters that will enter in our cost/objective function i.e. building cost and computational cost. Details of these parameters have already been discussed in section 4.2. Based on these parameters we have two objectives that are conflicting in nature. If we want to minimize building cost then computational cost will increase and vice versa. Hence we want to find a the balance between economy and computation efficiency.

Let us start developing cost function which is a bi-objective function:

Our first objective is to minimize the building cost. This can be written as:

$$\min \sum_{i} BC_i.N_i \tag{4.4}$$

where

- BC_i denotes the building cost of i^{th} component in the system.
- $N_i \in \{1, 0\}$ indicates if the i^{th} node is present in the system or not.
- $\sum_{i} BC_i N_i$ is the total building cost of all the components that are present in the SDR system.

Now let us consider our second objective i.e. to minimize the computational cost. This can be written as:

$$\min \sum_{k} CC_k \tag{4.5}$$

where

- CC_k denotes the computational cost of k^{th} component in the system.

Since we are concerned with the optimization of a multi-standards SDR systems, (4.5) can take the form:

$$\min \sum_{n} \sum_{k} CC_k((S_n)n \in N) \tag{4.6}$$

where

- $((S_n)n \in N)$ indicates that there may be n standards present in an SDR where $n = \{1, 2, \ldots, N\}$ i.e. if we choose $N = 3$ then it means that there are three standards namely S_1, S_2 and S_3 present in an SDR system.
- $\sum_{k} CC_k((S_n)n \in N)$ is the total computational cost of any of S_n, where $n = \{1, 2, \ldots, N\}$.

- $\sum_n \sum_k CC_k((S_n)n \in N)$ is the total computational cost of all of S_n, where $n = \{1, 2, \ldots, N\}$.

Combining the two objective function of equation (4.4) and (4.6) we get:

$$\min \Big(\sum_i BC_i.N_i + \sum_n \sum_k CC_k((S_n)n \in N) \Big) \tag{4.7}$$

These two objectives are not of equivalent importance so here we introduce some weights with two objectives. Obviously for some designers BC is more significant than other designers who are more concerned with the CC. Using these weighting factors (4.7) takes the form:

$$\min \Big(\bar{\omega} \sum_i BC_i.N_i + \sum_n \sum_k \omega_n CC_k((S_n)n \in N) \Big) \tag{4.8}$$

Where $\bar{\omega}$, is the weight given to the total BC of the system and ω_n is the weight given to the executing standard S_n once. It is assumed that these standards are not executing simultaneously.

A designer of multi-standard SDR system will specify the number of the standards to be supported. The solution for an optimal design consists in minimizing:

$$\mathbf{C_{SDR}} = \min_{bool((S_n)n \in N)} \Big(\bar{\omega} \sum_i BC_i.N_i + \sum_n \sum_k \omega_n CC_k((S_n)n \in N) \Big) \tag{4.9}$$

The constraint $bool((S_n)n \in N)$ therefore checks that all the standards are implemented in an SDR system.

Cost at any level of the graph (keeping in view multiple levels of graph) can be computed by

$$Cost_{level} = \Big(\sum_i CC_i.NoC_i \Big).NoC_{i+1} + \sum_i BC_i.N_i \tag{4.10}$$

Although the cost function seems to be simple one but the problem is very complex. There are many techniques that can be applied to solve (4.9). Our aim is to find a solution that balances between economy and computational efficiency as depicted in Fig. 4.2. Fig. 4.2 shows the solution space of our developed cost function. By using the optimization techniques we want to find a solution that minimizes the cost of our design. This minimum value of cost function will give us best compromise between BC and CC. It is to be noted that multiobjective cost function may reach a lot of different convergence local points which may not necessarily be desired global ones. Selected techniques of optimization and developed optimization tool are topic of the next chapter.

4.5 Graphical user interface

In this section we describe briefly the graphical user interface tool, developed to facilitate drawing of graph models and annotating the related cost parameters. The screen shot of the tool for drawing these graphs is shown in Fig. 4.3.

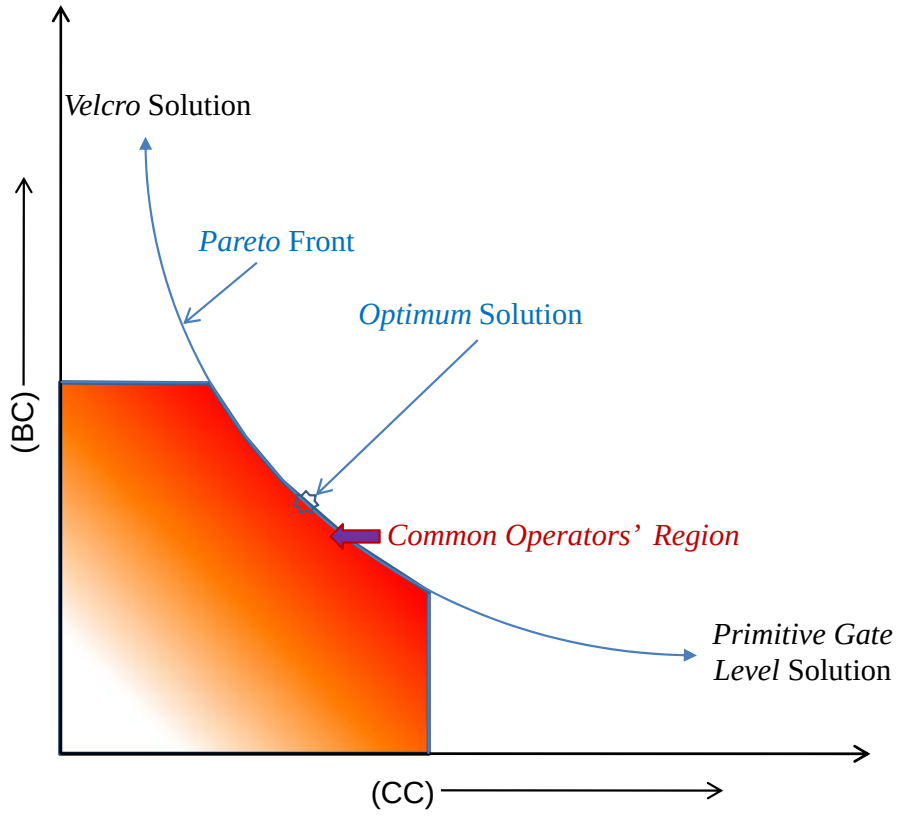

Figure 4.2: Solution space of cost function

The GUI of Fig. 4.3 has various buttons. Functionalities of some important buttons of the GUI are briefly explained below:

Create Processing Element: This button is used to draw processing elements. We can draw required processing elements, name them and position them accordingly, by using this button.

Delete Processing Element: This button is used to delete an already drawn PE.

Create AND Hyperarc: This button is used to create an AND hyperarc. The AND hyperarc is created by selecting a parent PE and respective children PEs. Using the dialog box this selection is made quite easily.

Create OR Hyperarc: This button is used to create an OR hyperarc. This hyperarc is created by selecting a parent PE and a child PE.

Delete Hyperarc: This button is used to delete any of created hyperarcs.

Add Parameters: This button is used to invoke *parameters's adjustment* window. Parameter's adjustment window is shown in Fig. 4.4. In our graph model we have two

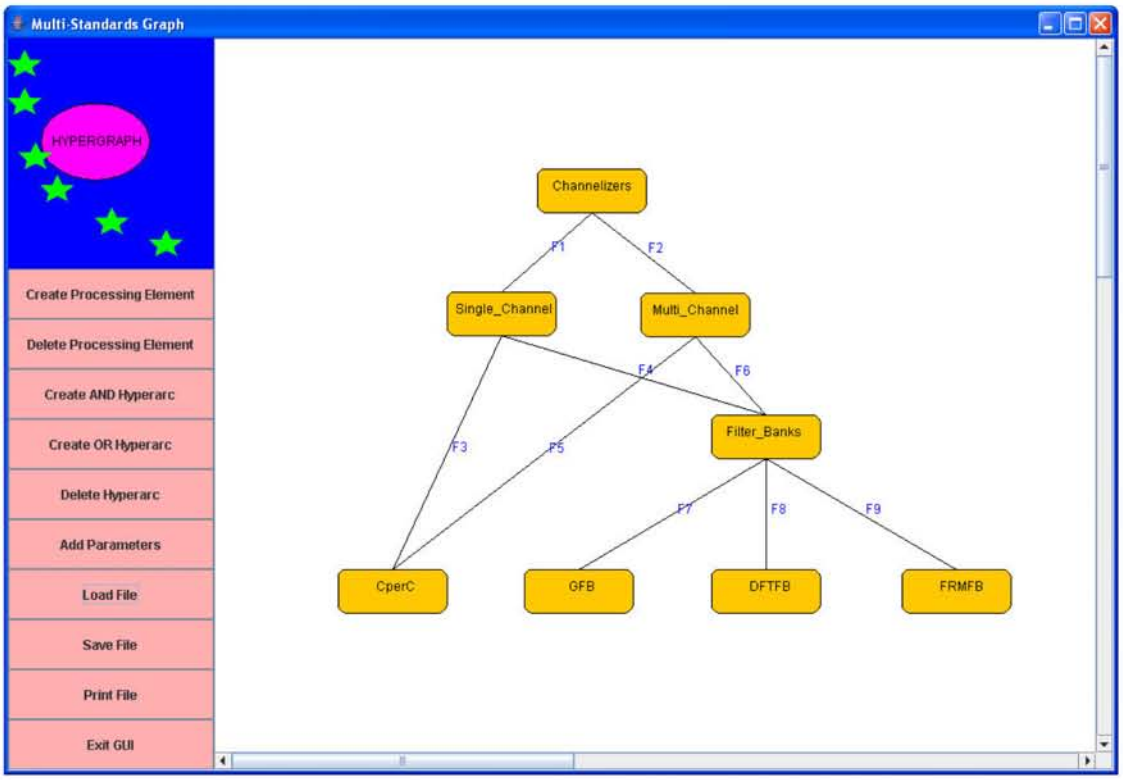

Figure 4.3: Screen shot of GUI for drawing graph models

entities i.e. processing elements (PEs)/blocks and hyperarcs (OR and AND). The purpose of this window is, firstly, to list all the parameters that can be associated with these two entities and secondly, to associate the appropriate parameters and their values with respective graph entities. All this is done by using this window. In this window we have the following functions that give us all the statistics of graph model and help us in parameters' adjustment.

- Parameters of Processing Elements
- List of Processing Elements
- Parameters of Hyperarcs
- List of OR hyperarcs
- List of AND hyperarcs

Parameters of Processing Elements, lists all the parameters that are associated with the processing elements. We can expand the drop down box in Fig. 4.4 to see all of these parameters. Here we have the options to add new parameters for the PEs, delete and/or rename existing parameters. *List of Processing Elements* gives a list of all the PE used to graph model the multi-standards SDR equipment. By using the drop down box in Fig. 4.4 we can see all the processing elements that are present in the graph model. We can select

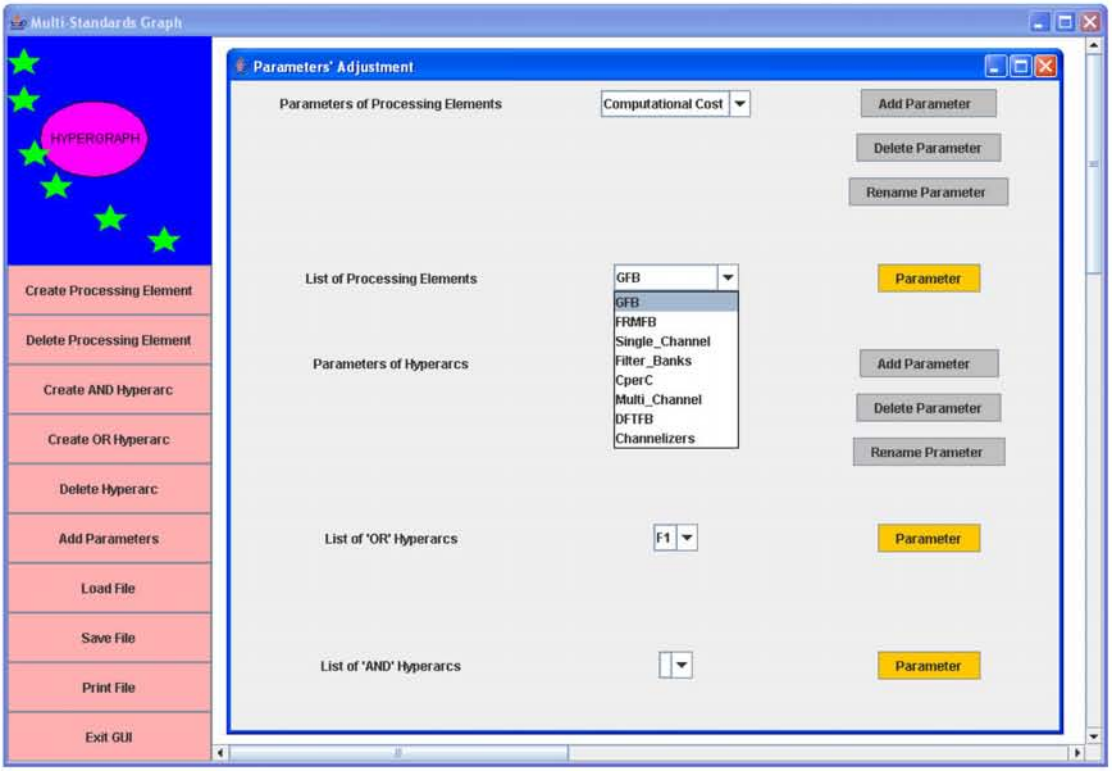

Figure 4.4: GUI with parameters' adjustment window

any of these PE and then by using *Parameter's* button located right next to selected PE, we can associate any parameter/s to this PE or delete any associated parameter/s. All the parameters are defined by using the *Parameters of Processing Elements*, and here, we only associate the appropriate parameters with the PEs from the already defined list. In addition to association we can adjust the value of respective parameters, e.g. selected *PE* is MAC, associated *parameter* is building cost and *value* is 20. Same procedure is repeated for other parameters of this PE and for all PE that are present in the graph model.

Parameters of Hyperarcs, lists all the parameters that are associated with the hyperarcs. We can expand the drop down box in Fig. 4.4 to see all of these parameters. Here we have the option to add new parameters for the hyperarcs, delete and/or rename existing parameters. *List of OR/AND hyperarcs*, lists all of OR/AND hyperarcs as evident from name. By using the drop down box in Fig. 4.4 we can have a look at all the OR/AND hyperarcs that are present in the graph model. We can select any of these OR/AND hyperarcs and then by using *Parameter's* button located right next to selected hyperarc, we can associate any parameter/s to this hyperarc or delete any associated parameter/s. All the parameters are defined by using *Parameters of Hyperarcs*, and here, we only associate the appropriate parameters with the hyperarcs from the already defined list. In addition to associating, values of these parameters are also adjusted as explained above in case of processing elements.

After drawing the graph and annotating the cost parameters, the graph is saved in a special format to be used by parser for our optimization tool. GUI tool also provides the options of loading and printing graph models.

4.6 Conclusions

We started our discussion with the fundamental question: how to select the best available alternative to design a multi-standard SDR system. We came to the conclusion that there is no other solution rather then turning graph into an optimization problem and apply the various established optimization techniques to find the best possible solution.

We discussed various cost parameters that can be associated with graph entities and various cost factors that can be attributed to these cost parameters. We also discussed the available hardware choices for SDR implementation and how these choices can effect our design and consequently the cost parameters.

We addressed the problem of formulating a cost function that is to be minimized in order to find a solution that is a best compromise between cost and computing efficiency. We were able to find a cost function that look fairly appropriate to our goals.

In the next chapter we are going to address various optimization techniques that are available to solve/minimize this cost function.

Chapter 5

Selected Optimizations Techniques

Contents

5.1 Introduction

In the previous chapter we formulated our bi-objective cost function. In this chapter we discuss different techniques to solve this cost function. We want to minimize this cost function. Our aim is to find a solution that balances between economy and computational efficiency as mentioned in previous chapter.

To determine a solution of our problem we have two choices; either we can use techniques that give exact optimal solution or we can use techniques that provide near-optimal solution in less computing time. All exact methods known for determining an optimal solution require a computing effort that increases exponentially with number of nodes, so that in practice exact solutions can be attempted only on problems involving fewer nodes.

We start with an overview of major optimization techniques available and discuss their pros and cons. Based on this exploration, we select some techniques that are most suitable to the problem at hand.

We describe the exhaustive/enumerative search method that computes values of all the feasible solution every time. Since the problem is NP-complete, this method becomes impractical for large number of nodes. However, we will use this method for simple cases, to validate the results given by the stochastic techniques.

To obtain a near-optimal solution we discuss two stochastic techniques in detail namely, simulated annealing (SA) and genetic algorithms (GA).

Annealing is the process of slowly cooling a physical system in order to obtain states with globally minimum energy. By simulating such a process, near globally-minimum-cost solutions can be found for very large optimization problems. Simulated Annealing is a random-search technique which exploits the analogy between the way in which metal cools and freezes into a minimum energy crystalline structure and search for a minimum in a more general system. SA and its implementation is discussed in section 5.2.2

A genetic algorithm is an iterative procedure that maintains a population of individuals and these individuals are candidate solutions to the problem being solved. Each iteration of the algorithm is called a generation. During each generation the individuals of the current population are rated for their effectiveness as solutions. Based on these ratings a new population of candidate solutions is formed using specific genetic operators as explained in section 5.2.3.

In 5.3 we present a generic design example. We apply selected optimization techniques to this generic example in order to find out that which of these techniques is most appropriate for the problem at hand. A conclusion's section ends this chapter.

5.2 Overview of general optimization techniques

General search and optimization techniques can be classified into two major categories:

- Deterministic
- Stochastic

Various optimization techniques that can be grouped into above categories are shown in Fig. 5.1.

Exhaustive/enumerative search finds the best global solution after checking each and every solution in the search space. Even if it finds the best solution at the very beginning of the search, the search duration is not reduced. This technique is simplest in its implementation but it is inefficient or even infeasible as search spaces become large. The time required to compute each and every possible solution, in case of large search spaces, will be large enough even with fastest worldwide computers. Size of the search space of real world problems is usually enormous.

Basic depth-first is *uninformed* in that the search order is independent of solution location (except for search termination). It expands a node, generates all successors, expands a successor, and so forth. if the search is blocked e.g. it reaches a tree's bottom level, it resumes from the deepest node left behind [164]. Backtracking is a depth-first search variant which backtracks to a node's parent if the node is determined unpromising [165].

Breadth-first search also fall in the category of uninformed search techniques. It differs from depth-first search in its actions after node expansion. It progressively explores the

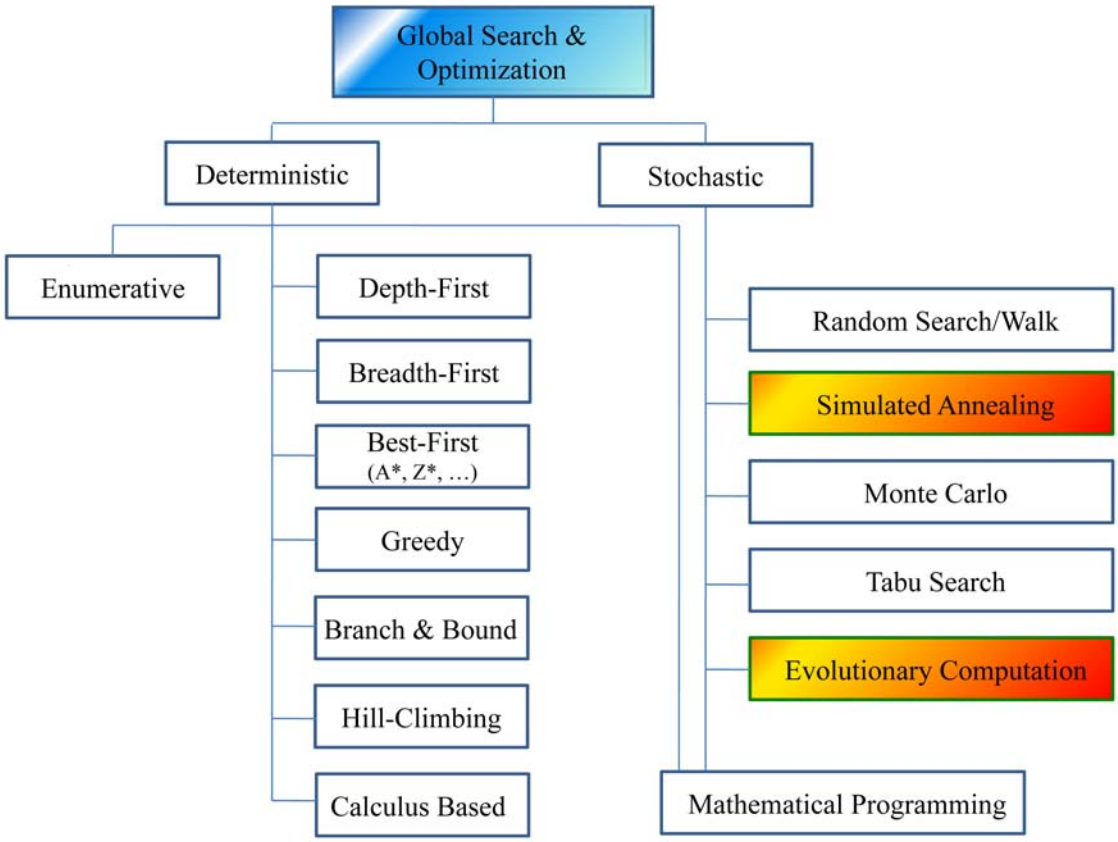

Figure 5.1: Global optimization approaches [35]

graph one layer at a time [164].

Best-first search uses heuristic information to place numerical value on a node's promise. The node with highest promise is examined first [164]. A^* and Z^* and others are popular best-first search variants selecting a node to expand and are based on promise and the overall cost to arrive at that node.

Greedy algorithms make locally optimal choices, assuming optimal sub-solutions are always part of the globally optimal solution [166, 167]. Thus these algorithms fail unless that is the case.

Hill climbing algorithms search in the direction of steepest ascent from the current position. These algorithms work best on unimodal functions, but the presence of local optima, plateaus or ridges in the fitness (search) landscape reduce algorithms' effectiveness [168]. Greedy and hill climbing strategies are irrevocable. They repeatedly expand a node, examine all possible successors then expanding the most promising node and keep no record of the past expanded nodes [164].

Branch and bound search techniques need problem specific heuristics/decision algorithms to limit search space [164, 169]. They compute some bound at a given node which determines whether the node is promising or not; then several nodes bounds are compared and the algorithm branches to the most promising node [165].

Finally calculus-based search methods at a minimum require continuity in some variable domain for an optimal value to be found [170].

Depth-and breadth-first search, best-first search, greedy and hill climbing algorithms, branch and bound tree/graph search techniques and calculus-based methods are all deterministic methods used in solving a wide variety of problems [165, 166, 171]. However, many multi-objective problems (MOPs) are high dimensional, NP-complete, discontinuous and/or multi-modal. Deterministic methods are often ineffective when applied to NP-complete or other high dimensional problems because they are handicapped to direct or limit search in these exceptionally large search spaces. Problems exhibiting one or more of these above characteristics are termed irregular [172].

Many real-world scientific and engineering MOPs are irregular and hence deterministic search techniques are not suitable for them. To solve these irregular problems, stochastic search and optimization approaches such as Simulated Annealing (SA) [173], Monte Carlo methods [174], Tabu search [175] and Evolutionary Computation Algorithms [171, 176, 177] were developed as alternative approaches. SA is a generalization of Monte Carlo methods for examining the equations of state and frozen states of n-body systems [178].

Stochastic methods require a function assigning fitness values to possible or partial solutions, and an encode/decode (mapping) mechanism between the problem and algorithm domains. Some of these methods are shown to eventually find the optimum solution but most of them cannot guarantee the optimal solution. In general they provide good solutions to wide range of problems which traditional deterministic methods find difficult as discussed in [171, 167].

A random search is the simplest stochastic search strategy, as it simply evaluates a given number of randomly selected solutions. A random walk is very similar, except that the next solution evaluated is randomly selected using the last evaluated solution as a starting point [179]. These strategies are not efficient for many MOPs because of their failure to incorporate problem domain knowledge. Random searches are generally expected to do no better than enumerative ones [171].

Simulated Annealing algorithm is explicitly modeled on an annealing analogy. Annealing is the process of slowly cooling a physical system in order to obtain states with globally minimum energy. By simulating such a process, near globally-minimum-cost solutions can be found for very large optimization problems. Where a hill-climbing algorithm chooses the best move from some node, SA a random one. If the move improves the current optimum it is always executed, else it is made with some probability $p < 1$ as explained in Fig 5.2. This probability exponentially decreases either by time or with the amount by which the current optimum is worsened [168]. If the metal temperature is lowered

slowly enough then metal cools and freezes into a minimum energy crystalline structure; the analogy for SA is that if the move probability decreases slowly enough then we will find the global optimum.

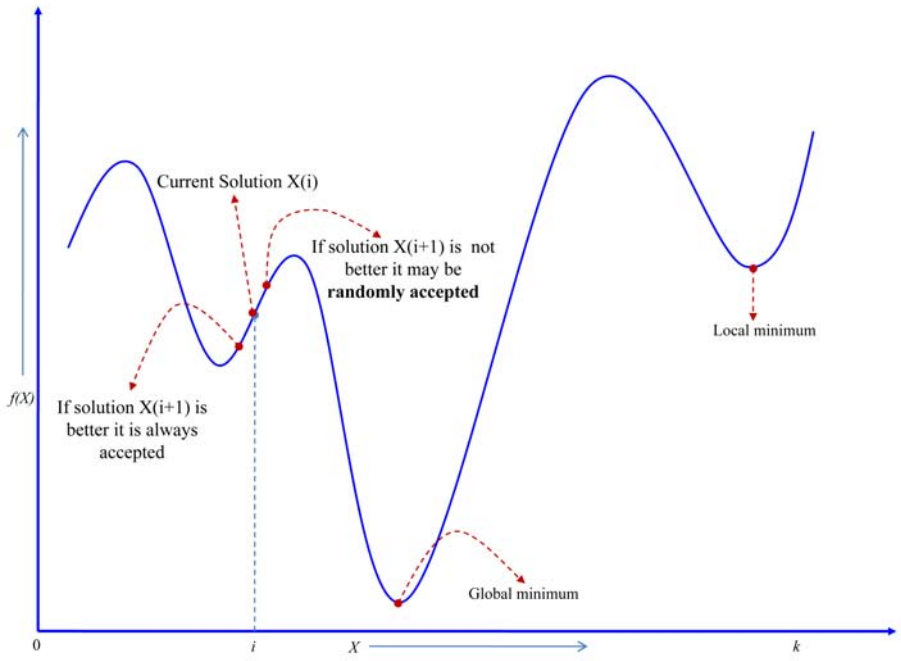

Figure 5.2: Principal of SA for the selection of new states

Monte Carlo methods involve simulations dealing with stochastic events. They employ a pure random search where any selected trail solution is fully independent of any previous choice and outcome [174, 180]. The current best and associated decision variables are stored as a comparator.

Tabu search is a meta-strategy developed to avoid getting stuck on local minima. Is keeps track of both visited solutions and the paths which reached them in different memories. This information restricts the choice to evaluate next. Tabu search is often integrated with other optimization methods [175, 180].

Evolutionary computation is a generic term for several stochastic search methods which computationally simulate the natural evolutionary process. Evolutionary computation embodies the techniques of genetic algorithms (GA), evolution strategies (ESs) and evolutionary programming (EP), collectively known as Evolutionary Algorithms (EAs). These techniques are loosely based on natural evolution and the Darwinian concept of *Survival of the Fittest* [171]. In general, an EA consists of *population* of encoded solutions, manipulated by a set of *operators* and evaluated by some *fitness function*.

5.2.1 Exhaustive search techniques

As the name implies, exhaustive search checks each and every solution in the search space until the best global solution has been found. That means if we do not know the value that corresponds to the evaluated worth of best solution, there is no way to be sure that we have found the best solution using exhaustive search unless we have examined everything. No wonder it's called exhaustive search. However, it is easily seen that this technique is inefficient or even infeasible as search spaces become large. Size of the search space of real world problems is usually enormous. It might require centuries or more of computational time to check every possible solution.

Exhaustive algorithms (also called enumerative) are interesting in some respects. At the very least, they are perhaps the simplest search strategy. The only requirement is to generate every possible solution to the problem systematically. There are ways to reduce the amount of work we have to do. The basic question to answer when applying an exhaustive search is: How can we generate a sequence of every possible solution to the problem? The order in which the solution are generated and evaluated is irrelevant because the plan is to evaluate every one of them. The answer for the questions depends upon the selected representation.

5.2.1.1 Implementation

The implementation of the algorithm is very simple. Fig. 5.3 shows its structure. We start by initializing nodes and cost. Initialization of nodes is done by sorting of nodes. These nodes are categorized as useful nodes, useless nodes, high nodes and low nodes. Useful nodes are directly implementable and have the necessary costs associated with them while useless nodes are non implementable. High nodes are stem nodes in the graph and low nodes are leafs. Once nodes are sorted, the total cost of the system is calculated based on useful nodes.

Primarily a set of nodes that consists of all the useful nodes in a graph is taken into account for finding the cost. Then a different set of useful nodes formed (by taking out/replacing back, one/many nodes out of total useful nodes); graph feasibility is checked and cost is updated. If the new cost is less than the initial cost it is accepted. Same procedure is done to check all the other alternatives having different set of nodes. Having checked all the possible alternatives we select the solution that gives minimum cost. This is the optimal solution with global minimum cost. Details of the coding methods/functions related to nodes, arcs, graph and exhaustive/brute force technique can be be found in Appendix C.

5.2.1.2 Design example

Let us consider the graph shown in Fig. 5.4. This figure shows the graph model of system that requires equalization, multichannel and OFDM. Different alternatives for the realization of these functions are shown. It is to be noted that only some block are tagged with the costs and some arrows are tagged with the number of call parameters. This is because of the fact the we want to run the exhaustive search to find the best solution and for running the exhaustive search the number of nodes must be small. So the number of PE/nodes considered with costs are less than 10. The nodes with the costs are designated

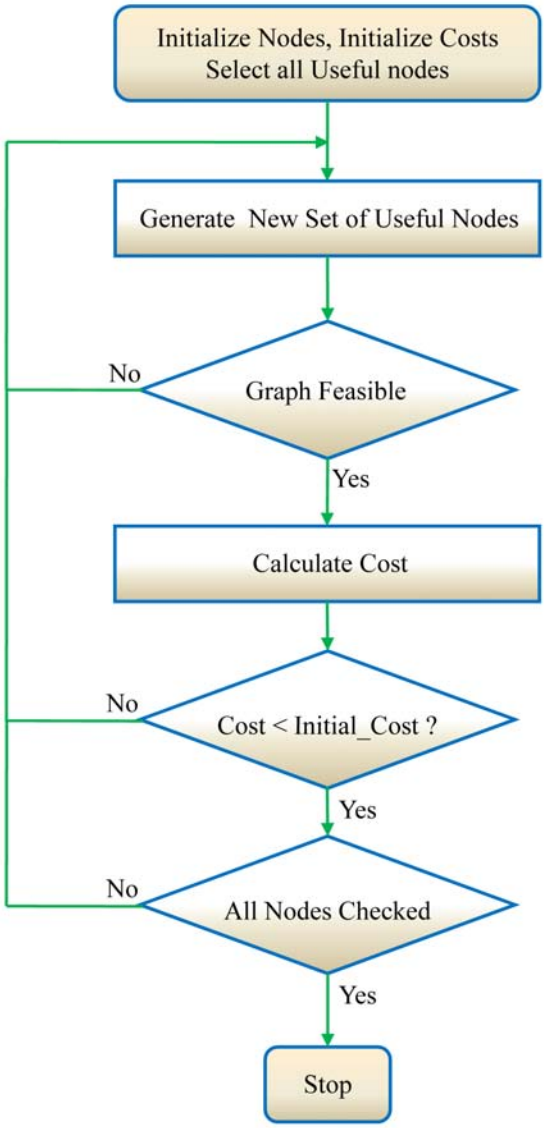

Figure 5.3: Flow diagram of exhaustive search algorithm

as useful nodes while sorting of nodes as mentioned earlier. The costs associated with the PE are BC/CC e.g. for Multiplier BC/CC=300/1.5. The costs associated with PEs are analytical costs. Having these costs we can run our optimization tool.

As an example let us consider the FFT node as it is obvious that it permits to implement all the three major communication blocks. May be other less obvious solutions exist but it is the role of program to find them beyond intuition. The cost of the FFT node at different levels of implementation can be computed by equation (4.10) duplicated below:

$$Cost_{level} = \left(\sum_{i=1}^{i=m} CC_i NoC_i \right) \times NoC_{i+1} + \sum_{i=1}^{i=m} BC_i.N_i \tag{5.1}$$

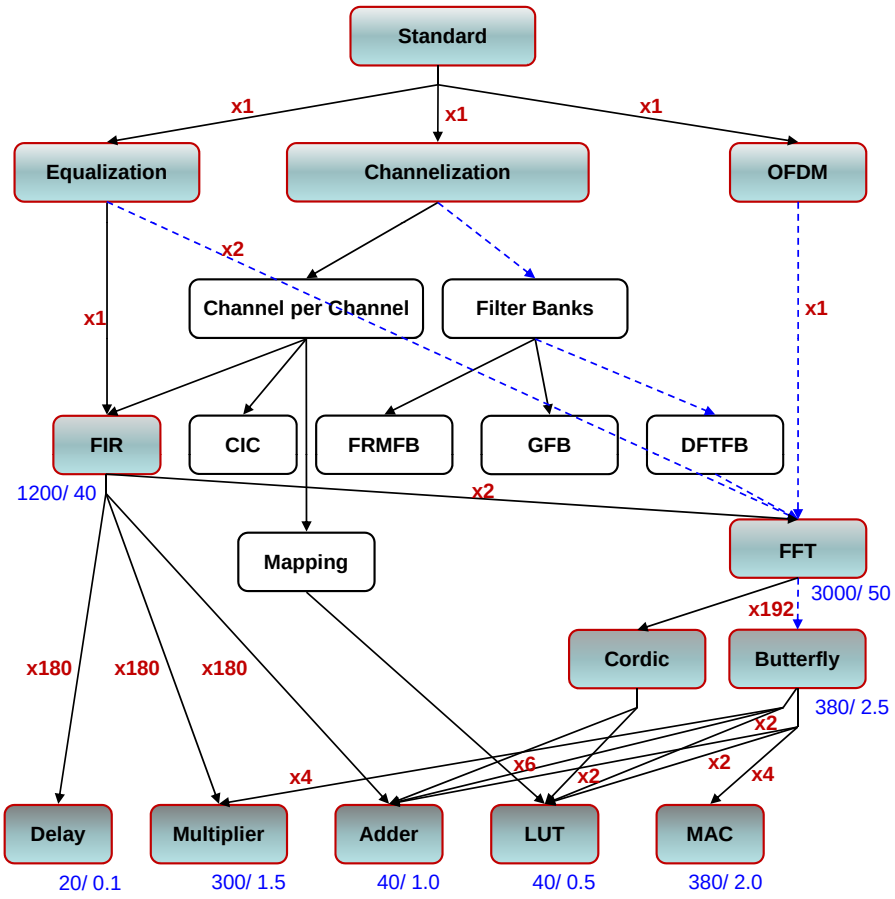

Figure 5.4: Graph model with associated costs

$N_i = 1$ since cost involves only those nodes that are present in the system. FFT is considered as the basic level in this example, hence $NoC = 1$ and, equation 5.1 becomes:

$$Cost_{FFT} = CC_{FFT} + BC_{FFT} \tag{5.2}$$

$$Cost_{FFT} = 50 + 3000 = 3050 \tag{5.3}$$

Let us use a level, lower than FFT to implement the functionality of FFT. Here we use Butterfly operator to implement FFT. The use of CORDIC operator for the FFT's implementation is not considered in this small example. As we are now at one level lower than FFT so NoC will have a value depending upon the number of points of FFT. If we consider the case of IEEE 802.11a WLAN [151] then $N = 64$ and $NoC = 192$ because for the implementation of 64-point FFT, we need to call a butterfly $(N/2) \times \log_2 N$ times and hence equation 5.1 becomes:

$$Cost_{\{FFT-1\}} = CC_{Butterfly} \times NoC_{Butterfly} + BC_{Butterfly} \tag{5.4}$$

or,

$$Cost_{\{FFT-1\}} = 2.5 \times 192 + 380 = 860 \tag{5.5}$$

We can further go down, lower than level of butterfly, to implement the functionality of FFT i.e. either we can use the operators: MAC, LUT and Adder or we can use the operators: LUT, Adder and Multiplier. In first case 4 MAC, 2 LUT and 2 Adders are required and in second case 4 Multipliers, 6 Adders and 2 LUT are required to build a Butterfly. A Butterfly is called 192 times to make the FFT. Building cost of each operator in either case is involved only once. However the computational cost is multiplied by the number of calls done at the operator to make run the global processing of system e.g. 192 calls of butterfly for the FFT with one occurrence of FFT to run the OFDM. Equation 5.1 now becomes:

$$Cost_{\{(FFT-2)\}} = \Big(\sum_{i=1}^{i=3} CC_i NoC_i\Big) \times NoC_{i+1} + \sum_{i=1}^{i=3} BC_i \tag{5.6}$$

Using the first option to implement the functionality of FFT, equation 5.6 becomes:

$$\begin{aligned} Cost_{\{(FFT-2)a\}} &= \big(4 \times CC_{MAC} + 2 \times CC_{LUT} + 2 \times CC_{Adder}\big) \times 192 \\ &+ BC_{MAC} + BC_{LUT} + BC_{Adder} \end{aligned} \tag{5.7}$$

or,

$$Cost_{\{(FFT-2)a\}} = \big(4 \times 2 + 2 \times 0.5 + 2 \times 1\big) \times 192 + 380 + 40 + 40 = 2572 \tag{5.8}$$

Using the second option to implement the functionality of FFT, equation 5.6 becomes:

$$\begin{aligned} Cost_{\{(FFT-2)b\}} &= \big(4 \times CC_{Multiplier} + 6 \times CC_{Adder} + 2 \times CC_{LUT}\big) \times 192 \\ &+ BC_{Multiplier} + BC_{Adder} + BC_{LUT} \end{aligned} \tag{5.9}$$

or,

$$Cost_{\{(FFT-2)b\}} = \big(4 \times 1.5 + 6 \times 1 + 2 \times 0.5\big) \times 192 + 300 + 40 + 40 = 2876 \tag{5.10}$$

Considering the design of equipment by FFT only we have the following cost function:

$$\begin{aligned} \mathbf{C_{FFT}} &= \min\big(Cost_{FFT}, Cost_{\{FFT-1\}}, Cost_{(\{FFT-2)a\}}, \\ &\quad Cost_{\{(FFT-2)b\}}\big) = \mathbf{Cost_{\{FFT-1\}}} \end{aligned} \tag{5.11}$$

Results of running the exhaustive search tool shows that Butterfly is the best common operator that implements the design with minimum cost for the graph of Fig. 5.4. Note that in the cost function, the weights $\bar{\omega}$ and ω_n are neutralized with 0.5 each and it will always be the cases thereafter, unless otherwise indicated.

5.2.2 Simulated annealing

Simulated Annealing (SA) is one of the most applicable metaheuristics in the optimization community. One of the most powerful features of SA is its ability of escaping from being trapped in local minima by accepting up-hill moves through a probabilistic procedure especially in the earlier stages of the search.

The algorithm is based upon that of Metropolis *et al.* [178], which was originally proposed as a means of finding the equilibrium configuration of a collection of atoms at a given temperature. The connection between this algorithm and mathematical minimization was first noted by [181], but it was Kirkpatrick *et al.* [173] who proposed that this algorithm forms the basis of an optimization technique for combinatorial (and other) problems.

The algorithm employs a random search, which not only accepts changes that decrease objective function, but also some changes that increase it. The latter are accepted with a probability given by equation 5.15, where $\Delta = [V(Y_k) - V(s)]$ is the increase in V and T_k is a control parameter, which by analogy with the original application is known as the system *temperature* irrespective of the objective function involved.

Suppose that a function V defined on some finite set S is to be minimized. We assume that for each state s in S that there is a set $N(s)$, with $N(s) \subset S$, which we call the set of neighbors of s. Typically the sets $N(s)$ are small subsets of S. In addition, we suppose that there is a transition probability matrix R over S such that $R(s, s^{'}) > 0$ if and only if $s^{'}$ is in $N(s)$.

Let $T_1, T_2, \ldots$ be a sequence (called a temperature schedule) of strictly positive numbers such that:

$$T_1 \geq T_2 \geq \ldots \quad and \quad \lim_{k \to \infty} T_k = 0 \tag{5.12}$$

Consider the following sequential algorithm for constructing a sequence of states $X_0, X_1, \ldots$. An initial state X_0 is chosen. Given that a potential next state is chosen from $N(s)$ with probability distribution as given in following equation:

$$P[Y_k = s^{'} | X_k = s] = R(s, s^{'}) \tag{5.13}$$

Then, we set

$$X_{k+1} = \begin{cases} Y_k & with \quad P_k \\ X_k & otherwise \end{cases} \tag{5.14}$$

where

$$P_k = \left[\frac{-[V(Y_k) - V(s)]}{T_k} \right] \tag{5.15}$$

$$\lim_{k \to \infty} P[X_k \in S^*] = 1 \tag{5.16}$$

Equation (**??**) specifies how the sequence $X_1, X_2, \ldots$ is chosen. The idea of the simulated annealing is to try to achieve (**??**) by letting T_k tend to zero as k tends to infinity, where S^* denote the set of states in S at which V attains its minimum value.

5.2.2.1 Implementation

The implementation of the algorithm is straightforward. Fig. 5.5 shows its structure. The following elements must be provided:

- A representation of possible solutions

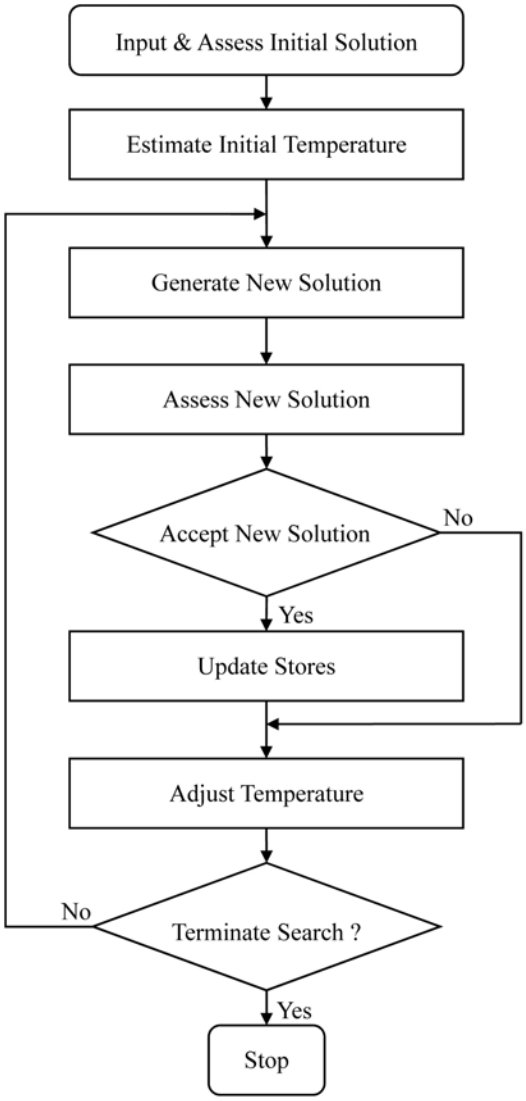

Figure 5.5: Flow diagram of simulated annealing algorithm

- A generator of random changes in solutions
- A means of evaluating the problem functions
- An annealing schedule - an initial temperature and rules for lowering it as the search progresses

The simulated annealing algorithm has been implemented by using C++. Details of the coding methods/functions related to nodes, arcs, graph and simulated annealing technique can be be found in Appendix C. Further details of the annealing algorithms for finite state-space (discrete time and continuous time) can be found in [173]. In the next section we apply the SA algorithm to solve the sub-graph as shown in Fig. 5.4 which has already been solved with exhaustive search technique in previous section.

Table 5.1: Results of SA at certain temperatures

Temperature $^{\circ}C$	**Selected PEs at different temperatures**	**Cost**
100	Delay, Multiplier, Adder, FIR, FFT	4760
70	FFT, Butterfly	3015
65	Butterfly	2200

5.2.2.2 Design example

For the design example we consider the same system used in 5.2.1.2. We start the SA algorithm with an initial temperature of $100^{\circ}C$. For this simple example, the function of decreasing temperature used is a linear function. Hence in each next step we decrease the temperature by $1^{\circ}C$, which is step size. First, a set of operators is selected at random by SA algorithm with a temperature of $100^{\circ}C$, graph feasibility is verified and cost corresponding to this set is calculated. After decreasing the temperature a new set of operators is selected at random by SA and cost for the new set is found. This procedure is repeated until a global minimum is found. Let us further explain the results of running SA algorithm (for certain temperatures) as given in Table 5.1. At $65^{\circ}C$, SA finds its minimum solution. However SA further lowers its temperature till $0^{\circ}C$ (stopping criteria) in order to avoid being trapped in local minima and to search for global minimum. At end the solution found at $65^{\circ}C$ appears to be global minimum. The results in Table 5.1 show the progress of SA algorithm. The procedure to determine the cost of a solution has already been detailed in section 5.2.1.2.

Results of running the SA algorithm show that selection of Butterfly as a common operator gives us the minimum cost. So our results of sub-optimal algorithm i.e. SA are in accordance with optimal solution.

5.2.3 Genetic algorithms

Genetic Algorithms are a family of computational models inspired by evolution. These algorithms encode a potential solution to a specific problem on a simple chromosome-like data structure and apply recombination operators to these structures so as to preserve critical information.

An implementation of a genetic algorithm [182, 171] begins with a population of (typically random) chromosomes. One then evaluates these structures and allocates reproductive opportunities in such a way that those chromosomes which represent a better solution to the target problem are given more chances to *reproduce* than those chromosomes which are poorer solutions. At each step, the GA selects individuals at random from the current population to be parents and uses them produce the children for the next generation. Over successive generations, the population *evolves* toward an optimal solution. The *goodness* of a solution is typically defined with respect to the current population.

GAs are often viewed as function optimizers although the range of problems to which genetic algorithms have been applied is quite broad. In general, a genetic algorithm is any population based model that uses selection and recombination operators to generate new

sample points in a search space. Many genetic algorithm models have been introduced by researchers largely working from an experimental perspective. Many of these researchers are application oriented and are typically interested in GAs as optimization tools. GAs can be applied to solve a variety of optimization problems that are not well suited for standard optimization algorithms, like problems in which the objective function is discontinuous, stochastic etc.

5.2.3.1 Basic terminology of genetic algorithms

In this section we explain some basic terminology for the genetic algorithms.

Fitness Function: The fitness function also know as objective function is the function we want to optimize.

Individuals: An individual also referred to as a chromosome is composed of entries called genes. The fitness function is applied to an individual.

Populations and Generations: A population consists of chromosomes. For example, in our problem we may use the population of 100 chromosomes. The same chromosome can appear more than once in the population. At each iteration, the genetic algorithm performs a series of computations on the current population to produce a new population. Each successive population is called a new generation.

Diversity: Diversity refers to the average distance between chromosomes in a population. A population has high diversity if the average distance is large; otherwise it has low diversity.

Fitness Values: The fitness value of a chromosome is the value of the fitness function for that chromosome.

Parents and Children: To create the next generation, the genetic algorithm selects certain individuals in the current population, called parents, and uses them to create individuals in the next generation, called children. Parents contribute their genes to their children. Typically, the algorithm is more likely to select parents that have better fitness values.

5.2.3.2 Rules of genetic algorithms

The genetic algorithm uses three main rules at each step to create the next generation from the current population:

1. *Selection rules* select the chromosomes, called parents, that contribute to the population at the next generation.

2. *Crossover rules* combine two parents to form children for the next generation.

3. *Mutation rules* apply random changes to chromosome parents to form children.

To understand these concepts, let us consider the canonical genetic algorithm (CGA) which was originally developed by Holland [182] as shown in Fig. 5.6. The key parts of Fig. 5.6 are described below in detail.

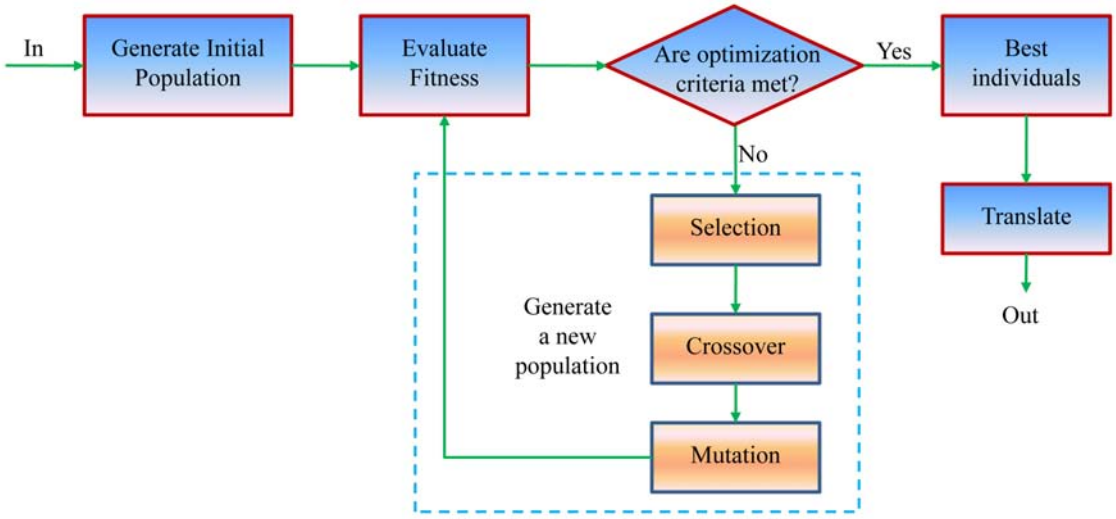

Figure 5.6: Canonical genetic algorithm

Initialization

The algorithm begins by creating a random initial population. To create the population, the coding structure is decided in the initialization phase. Coding for a solution, termed a *chromosome* in GA literature, is usually described as a string of symbols from {0, 1}. These components of the chromosome are then labeled as *genes*. The number of bits that must be used to describe the parameters is problem dependent. Let each solution in the population of m such solutions x_i, $i = 1, 2, \ldots, m$, be a string of symbols {0, 1} of length l. Typically, the initial population of m solutions is selected completely at random, with each bit of each solution having a 50% chance of taking the value 0. A simplified representation of population of chromosomes is shown in Fig. 5.7.

Selection

Proportional selection is used in canonical genetic algorithm. The population of the next generation is determined by n independent random experiments. The probability that an individual x_i is selected from the tuple $(x_1, x_2, \ldots, x_m)$ to be a member of the next generation at each experiment is given by the equation 5.17

$$P\{x_i \; is \; selected\} = \frac{f(x_i)}{\sum_{j=1}^{m}(f(x_i))} > 0 \tag{5.17}$$

This process is called *roulette wheel parent selection*. It may be viewed as a roulette wheel where each member of the population is represented by a slice that is directly proportional to the member's fitness. A selection step is then a spin of the wheel, which in the long

run tends to eliminate the least fit population members. Other selection techniques can be found in the literature [171, 183].

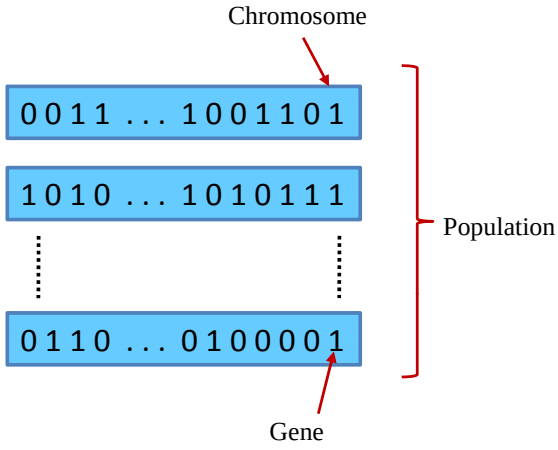

Figure 5.7: A simplified representation of population of chromosomes

Crossover

By using crossover rule we generate *child* chromosomes from two *parent* chromosomes by combining the *genes* extracted from parents. One point crossover method is used in CGA as shown in Fig. 5.8. In this method, for a chromosome of a length l, a random number r between 1 and l is first generated. The first child chromosome is formed by appending the last $l - r$ elements of the first parent chromosome to the first r elements of the second parent chromosome. The second child chromosome is formed by appending the last $l - r$ elements of the second parent chromosome to the first r elements of the first parent chromosome. Typically, the range of crossover probability is from 0.6 to 0.95. Crossover enables the algorithm to extract the best genes from different individuals and recombine them into potentially superior children. Different crossover techniques can be found in [171].

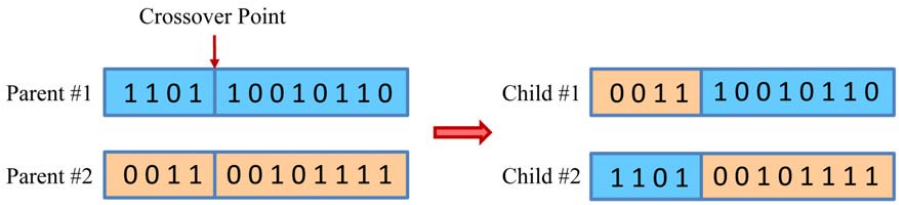

Figure 5.8: Crossover rule in CGA

Mutation

Mutation is another important rule in GA. This rule is applied independently on each individual by probabilistically perturbing each bit string. A usual way to mutate used

Figure 5.9: Mutation rule in CGA

in CGA is to generate a random number k between 1 and l and then make a random change in the k^{th} element of the string with probability $p_m \in (0, 1)$, as shown in Fig. 5.9. Typically, range of probability for bit mutation is from 0.001 to 0.01. Mutation adds to the diversity of a population and thereby increases the likelihood that the algorithm will generate individuals with better fitness values.
Without mutation, the algorithm could only produce individuals whose genes were a subset of the combined genes in the initial population. In addition to crossover children and mutation children as explained above, genetic algorithm creates another type of children called *elite children*. Elite children are the individuals in the current generation with the best fitness values. These individuals automatically survive to the next generation.

We can specify how many of each type of children the algorithm creates as follows: Elite count, specifies the number of elite children. Crossover fraction/probability, specifies the fraction of the population, other than elite children, that are crossover children. For example, if the population size is 100, the elite count is 20, and the crossover probability is 0.8, the numbers of each type of children in the next generation will be; There are 20 elite children. There are 80 individuals other than elite children, so the algorithm uses 0.8*80 = 64 to get the number of crossover children. The remaining 16 individuals, other than elite children, are mutation children. In this way we create the next generation of 100 chromosomes.

5.2.3.3 Implementation of CGA

In this section we describe the steps to implement the canonical genetic algorithm.

1. Create 10 chromosomes, randomly of 8 genes.
2. During random creation take care of repetition of combination of genes.
3. Distinct 10 chromosomes, randomly of 8 genes.
4. Check either the solution presented by a chromosome is valid or not (invalid();)
5. Calculate cost for each solution by chromosome.
6. Best chromosome (only the best one). For elitism save the best solution of current population.
7. Apply roulette wheel parent selection to create a parent population of 5 chromosomes.
8. Apply one point crossover with the probability rate of 80%.

9. Apply mutation to change the status of each gene.
10. Apply step 4 & 5 on 5 child chromosomes and retain valid children.
11. Combine children with previous population and select best 10 chromosomes to keep population constant.
12. Compare previous best to current best and save the current best.
13. Repeat step 7 to 13 for 100 generations or till the stall condition is reached.

Results of dry running the CGA on generic design example are provided in section 5.3

5.3 Application

In this section we apply the three selected optimization algorithms i.e. ES, SA and GA, on a generic example and try to evaluate the results to show which of the three technique is most suited to our problem.

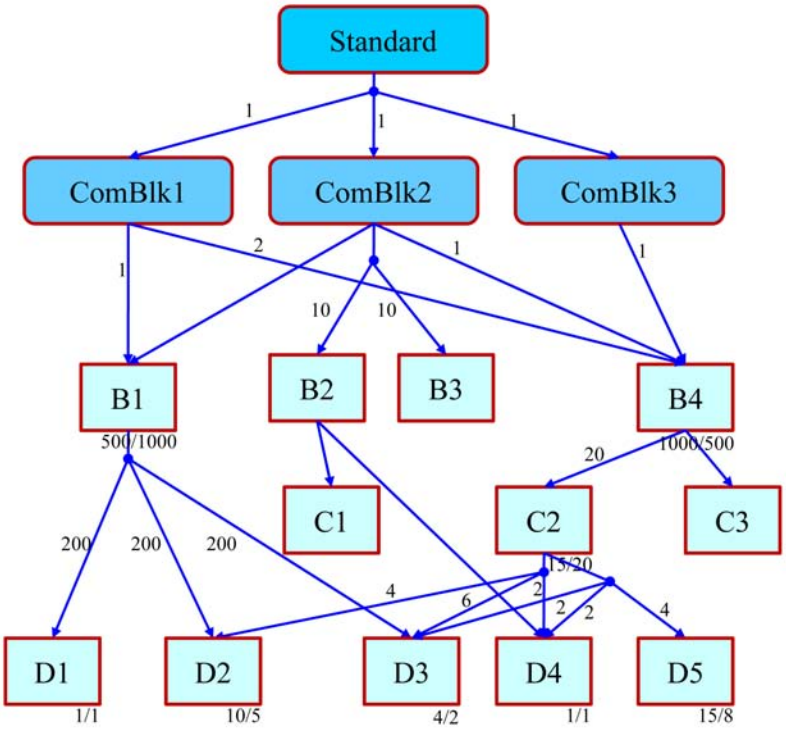

Figure 5.10: A generic view of SDR equipment

Let us consider a radio system as shown in the Fig. 5.10. This system consists of three major processing elements namely ComBlk1/2/3 and several small processing elements like B1, B2, ... C1, C2, ... and D1, D2, ... For running the optimization algorithms on system of Fig. 5.10 we need to associate cost parameters with PEs and arrows as mentioned earlier. Parameters that are associated with PEs are building cost/execution cost

(BC/EC). Parameter that is associated with arrows is number of calls (NoC). So we put values of all the parameters related to the entities of the graph as shown in Fig 5.10. It is to be noted that the values of all parameters are chosen arbitrarily and these values are not the real ones. In this section our purpose is to apply the optimization techniques as discussed above on the system of Fig. 5.10 and to find which one is best, so we consider here a generic system and not the real one.

Now as we are to compare the results of running ES, SA and GA. In order to run these algorithms on the system of Fig. 5.10, we are to restrict the number of PE; firstly because of the fact that if we take large number of PEs into consideration then it will be very difficult to run ES as it will take quite long time and, secondly, the comparison of three techniques will be very difficult because of large variations in the costs of solutions having large number of PEs. The restriction imposed on total number of PEs helps us to determine the suitability of one technique than others and it does not effects the generality of Fig. 5.10.

In order to restrict the number of PEs, we put the costs of only 8 PEs and in this way we declare the nodes with the costs as useful nodes. This serves our purpose of running and comparing the results of three algorithms.

Results of running the ES on design example of Fig. 5.10 are shown in Fig 5.11. Fig 5.11 shows the number of iterations versus costs of solutions. In the Fig. 5.11 all the solution whether good or bad are shown. We say that solution is good one if its cost is less as compared to the previous solution found, otherwise we call it a bad solution. The solid bars in Fig. 5.11 show the good solutions and dotted line shows the general optimization trend. Fig. 5.11 also shows that we find the minimum around 200th iteration for ES after checking all the possible solutions in case of ES.

Let us make the results of Fig. 5.11 more clearer. In Fig. 5.12 we show only the good solutions. By good solution we mean the solution that gives lower cost as compared to previous good one as mentioned earlier. Hence, Fig. 5.12 shows the bar graph of good solutions only. The dotted line shows the optimization tendency.

Now we run the SA algorithm on the same example of Fig. 5.10. Fig. 5.13 shows a bar graph of number of iterations versus costs of solutions in case of SA algorithm. It is evident from Fig. 5.13 that number of iterations required to reach the global optimal solution is much less as in the case of ES. Global minimum is found at 34th iteration in case of SA as can be seen in Fig. 5.13 while ES founds the same solution at 200th iteration. It is worth noting that SA outperforms ES in this comparison.

Results of dry running the CGA on generic design example are shown in Fig. 5.14. The population size of only 10 chromosomes is taken to facilitate the calculations by hand. Normally the population size are much higher e.g. 100, 200 etc. Crossover probability is assumed to be 0.8. Further the algorithm is dry run for only 10 generations for the reasons described above. In the Fig. 5.14 only *elite* (fittest) solution in each population is shown and not the complete population (10 chromosomes) at each generation. The purpose of

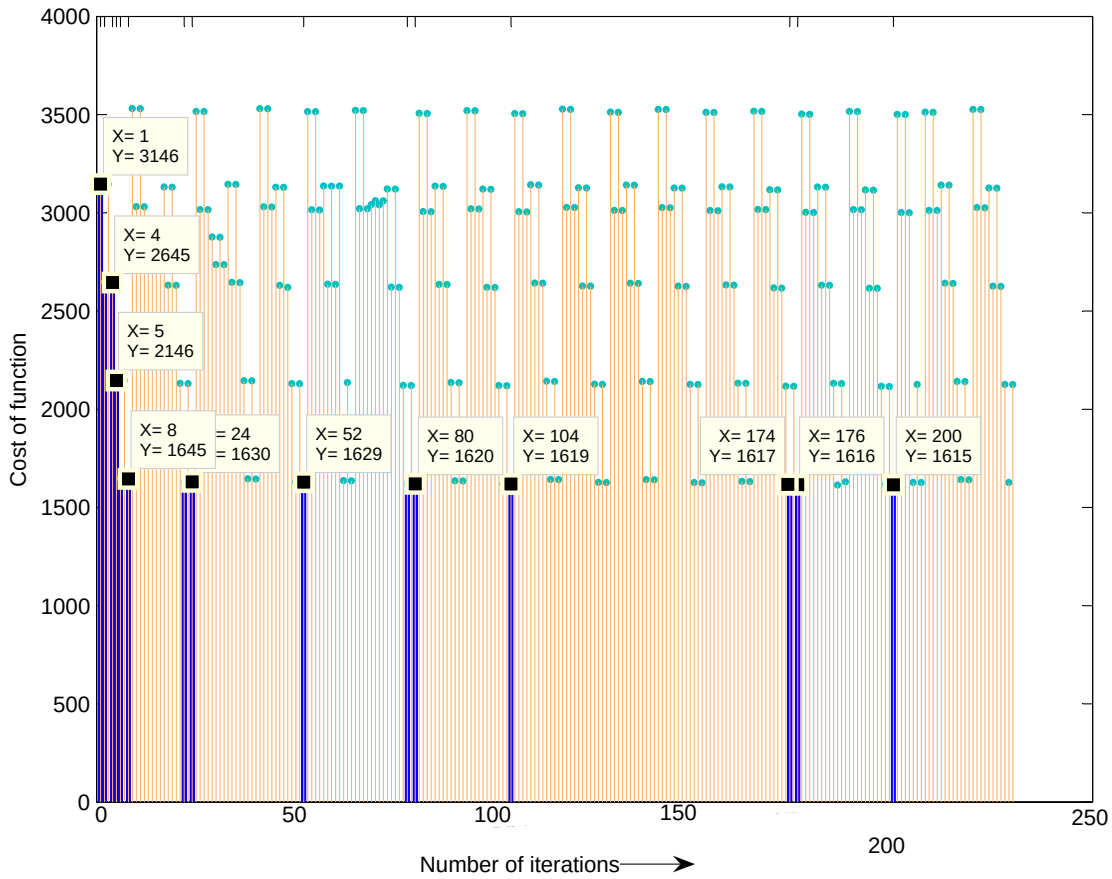

Figure 5.11: Results of running ES on generic design example

doing all this is to illustrate the GA running on Fig 5.10. The complete procedure for GA has already been outlined in 5.2.3.

It is to be noted that GA has been dry-run to a very small population size as it is extremely difficult to run the GA without its implementation on computer. The implementation of GA, its rules and determining the decision criteria for the rules of GA is a PhD topic in itself and therefore out of the scope of this thesis. However we present it here for the comparison purposes only.

Let us give another view of solutions versus iteration numbers. In Fig.5.15 we give a pictorial view of processing elements retained (that constitute the solution) versus number of iterations. In Fig. 5.15 we show only the good solutions at different number of iterations. Results show that if we implement the system with C2 then it will give us the least cost.

Cost mentioned at the top of each solutions in Fig. 5.15 is not the BC or EC alone. In fact this is the total cost that includes the BC and execution cost. The procedure to calculate these costs has already been explained in section 5.2.1.2. Fig. 5.15 gives an idea of cost of the system, number of iterations and number of processing elements contributing to

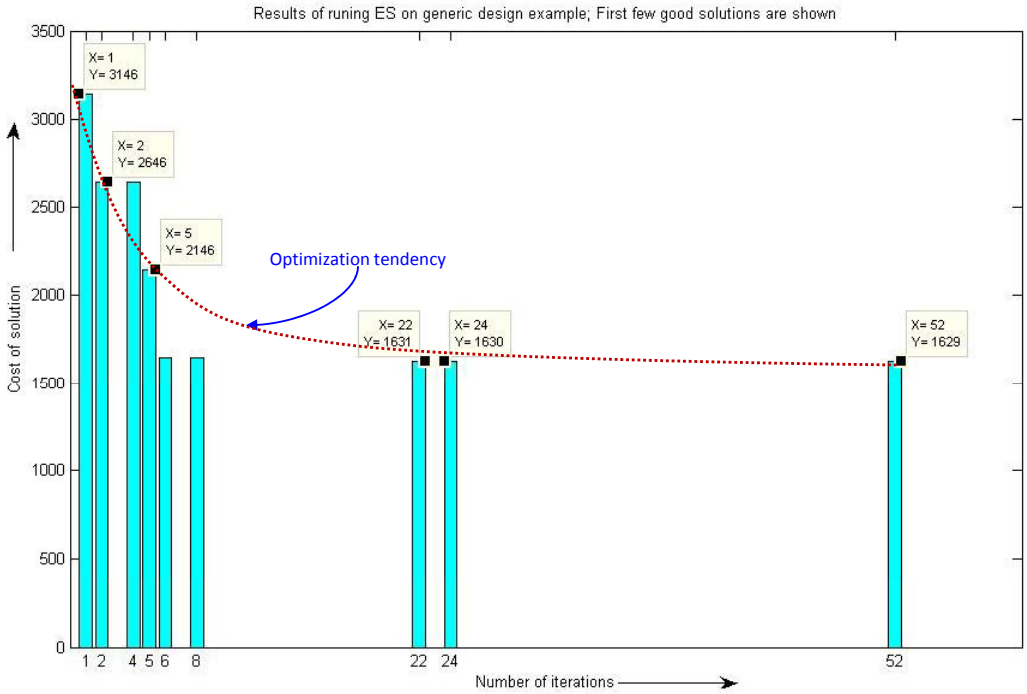

Figure 5.12: Results of running ES on generic design example with good solutions

that cost at the same time. Of course this is very small example but the purpose is to completely elaborate our idea and choose the appropriate technique. The real system will have huge number of PEs and in that case we have to use SA or GA as ES is not feasible considering the number of iterations and consequently the time required to evaluate the complete solution space and find the global optimal in case of ES.

Discussion

SA can be thought as GA where the population size is only one [36]. The current solution is the only individual in the population. Since there is only one individual, there is no crossover, but only mutation. This is in fact the key difference between SA and GA. While SA creates a new solution by modifying only one solution with a local move, GA also creates solutions by combining two different solutions. Both SA and GA share the fundamental assumption that good solutions are more probably found near already known good solutions than by randomly selecting from the whole solution space. The relative weight given to mutation and recombination is a crucial parameter affecting what a GA actually does. If mutation is the dominant way of creating new solutions, then the GA in fact acts as a parallelized version of SA, where several solutions are being independently improved. For some problems, evaluating solutions that are near an existing solution may be very efficient, which may give a big performance advantage to SA, when compared to

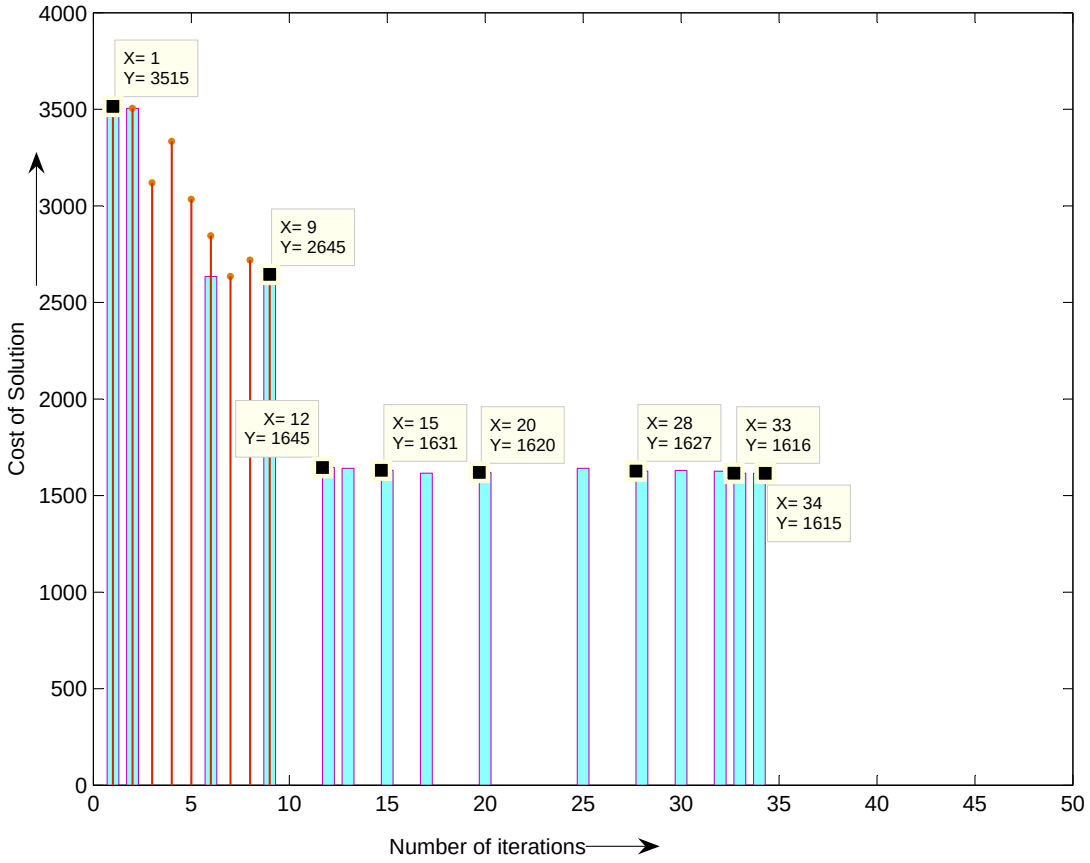

Figure 5.13: Results of running SA on generic design example

GA, if evaluating recombined solutions is not so efficient.

The execution time to determine the optimal solution which is a direct consequence of number of iterations is quite important in comparisons of optimization algorithms. If there were no limits on execution time/number of iterations, one could always perform a complete search, and get the best possible solution. As in case of generic design example, the number of iterations required by the SA algorithm is much less as compared to the ES. In the case of GA we can foresee that the stall condition will reach at higher number of generations as compared to number of iterations required in SA algorithm. Moreover, for a fair comparison (in case of GA) we need to convert number of generations into number of iterations. As population size taken for each generation is 10 chromosomes so we multiply number of generations with number of chromosomes to get an equivalent value in terms of number of iterations. Therefore as result we can say that SA algorithm outperforms the ES and GA in case of our generic design example. Hence SA algorithm is our choice for designing multi-standard SDR equipment. Let us present some examples from literature advocating the advantages of SA over GA, even if we cannot arbitrarily say that SA is better than GA in general. It depends on each optimization problem.

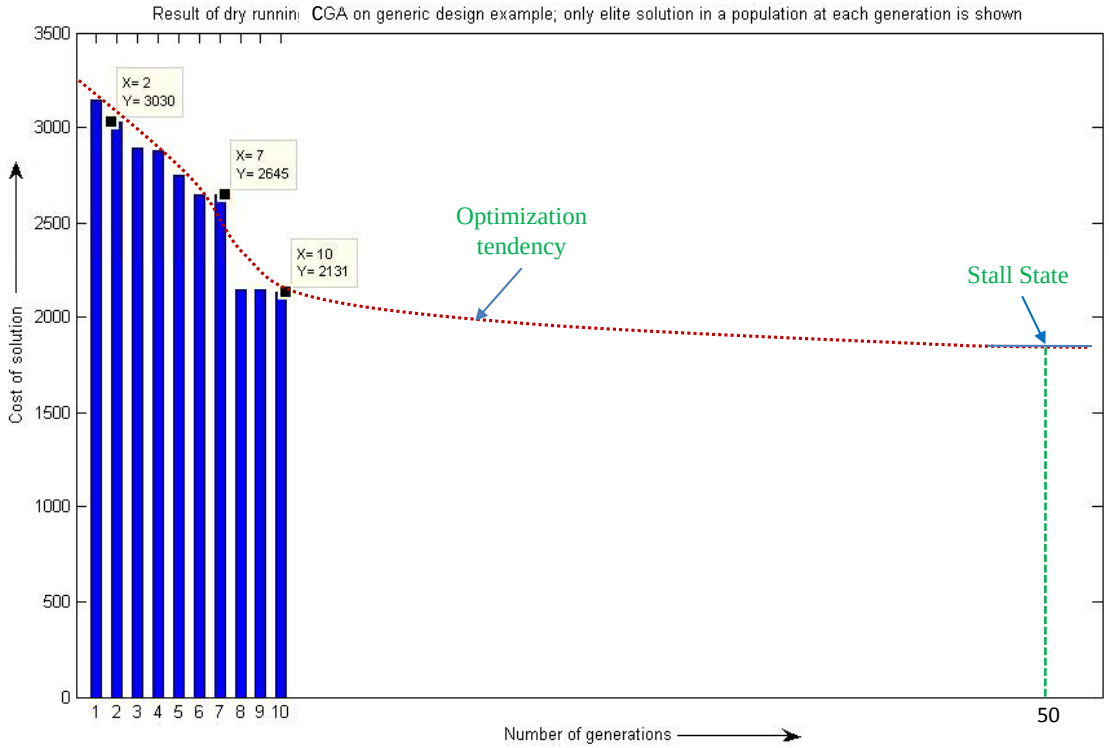

Figure 5.14: Results of running CGA on generic design example

In [37], a comparison of several algorithms including SA and GA for a tree cost minimization problem with carefully normalized execution time given to different algorithms is presented. Results indicate that given the same amount of time, SA consistently gave better solutions than GA.

In application of SA and GAs to the reconstruction of electrical permittivity images in capacitance tomography it is stated that memory allocation is much higher for the GA method because it requires the storage of a large binary coded matrix of models population [40]. SA gets slightly better results at the end of the whole inversion process.

In a comparison of the GA to SA algorithm in solving transportation location-allocation problems with euclidean distances it was shown that in all cases, the algorithm based on Simulated Annealing was superior to the other techniques [38]. The results illustrate that, the two tiered hybrid SA and Linear Programming (LP) method is better than the two tiered hybrid GA and LP method for solving transportation location problems.

Direct comparisons have been made between adaptive simulated annealing (ASA)/very fast simulated re-annealing (VFSR) and publicly-available genetic algorithm (GA) codes, using a test suite already adapted and adopted for GA [39]. In each case, ASA outperformed the GA problem. GA was not originally developed as an optimization algorithm, and basic GA does not offer any statistical guarantee of global convergence to an optimal point.

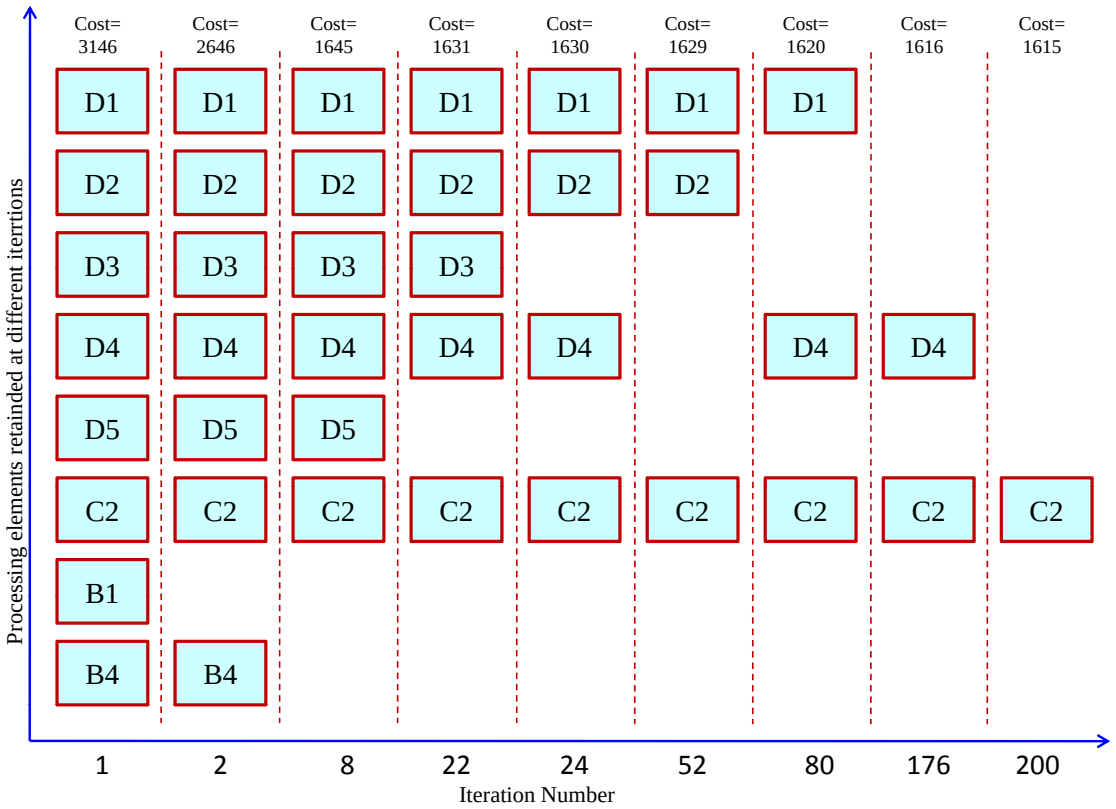

Figure 5.15: Processing elements retained at different iterations

Nevertheless, it should be expected that GA may be better suited for some problems than SA.

5.4 Conclusions

We started our discussion with an overview of general optimization techniques. We briefly mentioned the advantages and disadvantages of each of these techniques. Due to the limitation on number of nodes in case of optimal search methods we considered two sub-optimal techniques to solve our problem, namely, simulated annealing and genetic algorithm.

We discussed in detail the exhaustive search technique and its implementation using a small sub-design example. The evaluation of cost function at different levels of graph model was described. Then we addressed the simulated annealing algorithm and its implementation. Following this, the genetic algorithms and guidelines for their implementation were provided.

A generic design example was developed in order to compare the three techniques i.e. exhaustive search, simulated annealing and genetic algorithm. In addition to a bibliographical study, results of the comparison in small generic example presented in this

chapter show that among the three, simulated annealing is the best choice to solve our optimization problem. This is not a definitive conclusion since GAs are very convenient for problems involving large number of parameters and when no other solution appears to be feasible. However, this was not the case for the problem at hand. As expected SA out performs ES because it finds the optimum solution in much less number of iterations than ES. May be the work for the selection of best optimization technique could have been extended, but anyway this selection perfectly matched our needs for different case studies as presented later in this manuscript. Moreover, we are quite confident on the pertinence of our choice for much more complex graphs than the ones we faced. It is worth mentioning here that ES is not an algorithm studied to be used as candidate solution. ES (with fewer nodes) is used for the verification and comparison of the results obtained from other techniques.

In the next chapters we are going to address various common operators developed to be used in SDR equipment and apply the selected optimization techniques on sub-parts of real world SDR systems.

Part III

Introduction to Part III

In previous chapters we discussed the idea of common operators and their use in the design of multi-standards SDR systems. It was shown how the design problem can be modeled as graph. We also described various optimizations techniques to obtain optimal and near optimal solutions to design these systems. In the forthcoming chapters we are going to apply the discussed ideas on sub-parts of real world systems.

One aim of this thesis is to investigate/propose new common operators. Hence in this part we present three potential candidates for *common operators* to be used in multi-standards SDR systems and show how these CO can play their role in optimization of radio systems. This is one of our major contribution to the knowledge base. We describe these operators namely *Dual Mode Fast Fourier Transform* (DMFFT) operator, *Linear Feedback Shift Register* (LFSR) operator and *Frequency Response Masking Filter Banks* (FRMFB) operator, in detail and explain their respective architectures and various possible implementations strategies. We introduce some design scenarios and draw graphs for these scenarios. We integrate these operators in graphical models of aforementioned scenarios and run the optimization algorithms. Our ultimate aim is to integrate all the CO available in complete system graph and run the optimizations on it. However for the purpose of clarity each of the next chapters is focusing on one of the common operator and in each chapter a sub-design scenario is presented focusing around one CO. Results of running these algorithms highlight the advantages that we may have by using these operators in future SDR systems.

The research work described in this part is a result of collaboration with different Ph.D. student of our research team working in different labs. Each of these students has concentrated his research efforts on one CO study and its related necessary implementation on FPGA target technology. We feel privileged having been part of these collaborations and, having worked with different professors and fellow students from different labs.

The novel design of reconfigurable Fourier transform operator over $\mathbb{C}$ and $GF(F_t)$ is a result of combined effort with a post doctoral fellow working in Lab-STICC, University of South Brittany, Lorient, France. Chapter 6 provides all details regarding this work.

Development of LFSR operators to be used as common operators is the result of a joint work between us and a Ph.D fellow in CEA, Grenoble, France. Chapter 7 presents a complete description of this work.

Use of FRMFB as common operators is a joint effort between our laboratory and Nanyang Technological University, Singapore. Details of this work are given in chapter 8.
Our idea is to develop more and more complete graphs involving more and more COs and then find an optimized solution for designing a multi-standards equipment. In order to explain this we present a case study of a complex graph. In this graph we try to merge the COs presented in this part of thesis to find a global optimal solution. Details of this study are provided in chapter 9.

Chapter 6

Case Study: FFT as a common operator in multi-standard SDR systems

Contents

6.1 Introduction

The discrete Fourier transform (DFT) and the algorithms for its fast computation, the fast Fourier transform (FFT), are the best known and probably the most important in area of spectral analysis as they permit adequate representation in the frequency domain for all but the shortest of data lengths. Fourier analysis has been in existence since its publication in 1822 by Fourier [41] and has therefore achieved a higher degree of familiarity, respectability and development, as well as wide range of applications like telecommunications, medical electronics, seismic processing, radars, etc. This area was started in digital signal processing (DSP) with the publication of the Cooley-Tukey algorithm [42] which allows to reduce the order of complexity of some crucial computational tasks like complex multiplications from N^2 to $N \times log_2 N$ and complex additions from $N \times (N-1)$ to $N \times log_2 N$, where N is the transform length.

Starting from the useful considerations of the FFT in several and important steps of digital communications, the authors in [17] have considered the FFT as *common operator* (CO) and have detailed the various tasks which can be performed using the FFT operator as

mentioned earlier. Discussion in [17] implies that the FFT could become a CO for a terminal supporting several standards where the OFDM modulation and frequency domain algorithms are involved.

In this chapter, the channel coding (particularly Reed Solomon (RS) codes) in frequency domain which is of significant interest in many wireless systems (because the FFT operator is already present, at least in OFDM based ones) is examined. We draw an analogy between this frequency processing and the classical frequency signal processing based in the domain of complex field. Considering the performance comparison given in section D.2.3 of Appendix D, we propose to use RS codes defined over $GF(F_t)$ instead of RS codes over $GF(2^m)$. Based on this proposition we design a reconfigurable architecture of a common FFT operator able to operate over two different domains: $\mathbb{C}$ and GF. In this way, the designed FFT operator can be used in two different contexts: channel coding and any communication task requiring the FFT-$\mathbb{C}$ functionality as shown in the Fig. 6.1.

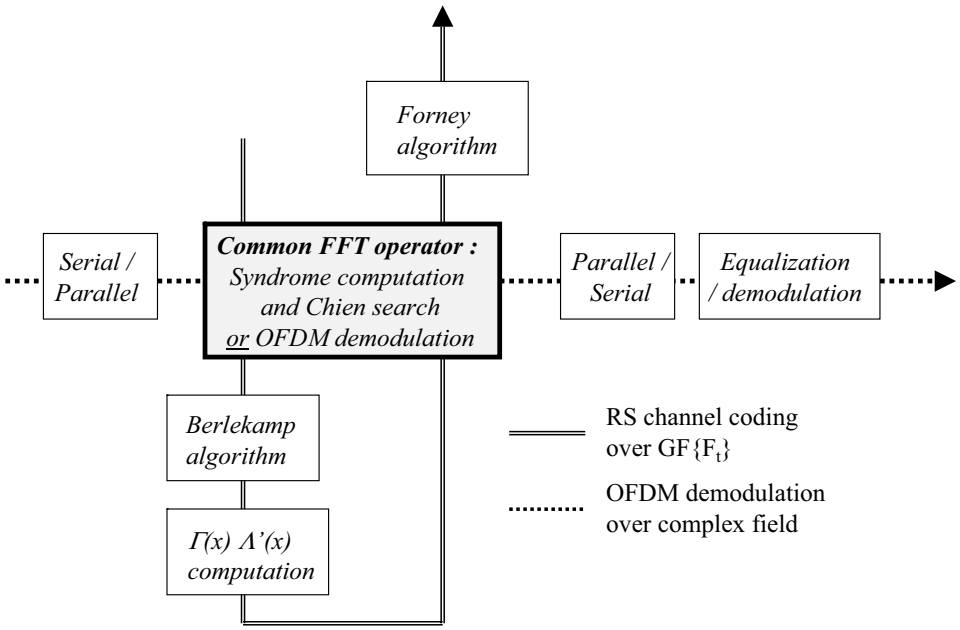

Figure 6.1: An example of FFT sharing between OFDM demodulation and RS decoding over $GF(F_t)$.

To support multi-standards in SDR, this operator called dual mode FFT (DMFFT) operator is able to reconfigure its hardware for the new standard/mode of operation. Various implementation strategies for DMFFT operator are discussed in section 6.3. The general architecture of DMFFT operator and architectures of its stages can be found in section D.3 of Appendix D. The architecture of core of DMFFT operator i.e. reconfigurable butterfly processing element (RBPE) is explained in same section. In this section we also present a complexity comparison of DMFFT operators with Velcro approach. These operators are implemented on Altera's Stratix-II FPGA family. A brief overview of Stratix-II FPGA

family is given in section D.4 of Appendix D. Results show that using DMFFT we gain in terms of Adaptive Lookup Table (ALUTs), memory and efficiency to perform FFT. We try to prove the same result by integrating this operator in our graph and running the optimization. In section 6.4, we present a design scenario that explains our approach to design multi-standard systems. We incorporate the DMFFT operator in our graph as mentioned earlier and solve optimization problem by selected optimization techniques discussed in chapter 5. Using results we infer that this DMFFT operator is a strong candidate for *common operators* in future air interface standards. The section of results and conclusions ends this chapter.

6.2 Channel coding and fast Fourier transform

In real world communication, errors are introduced in messages sent from one point to another. The received signal is always affected by various kinds of parasites, Gaussian or non-Gaussian noise, interference, fading, dispersion, etc. During the last decades since the rise of data-transmission, coding techniques have been necessary in digital communications and storage devices to reduce the error rate in the received data stream and improve the efficiency of the information transmission. RS is an error correcting coding system that was devised to address the issue of correcting multiple errors especially burst-type errors in mass storage devices [184], wireless and mobile communication units, satellite links, audio systems [185], High-Definition (HD) TV [186] etc. Reed Solomon (RS) codes are one of the most promising codes that are being adopted by many wireless standards like WiMAX, xDSL etc.

The choice of most important classes of cyclic codes, the RS codes is based on our motivation to investigate the channel coding in the frequency domain in order to find a way to implement some steps of channel coding with FFT. The originality of our work does not lie in the discovery of the frequency processing of the cyclic codes but the idea is to highlight the use of the Fourier transform for channel coding to prove that it is worth to consider FFT as a common operator. We emphasize the benefit of such a transform on the encoding and decoding processes of RS codes. The codes used for channel coding are defined in a less familiar domain known as Galois Field (GF).

Application of the discrete Fourier transform in the complex field occurs throughout the subject of signal processing. Fourier transforms also exist in the Galois Field $GF(q)$, and can play an important role in the study and processing of $GF(q)$-valued signals. By using the Fourier Transform, the idea of coding theory can be described in a setting that is much closer to the methods of signal processing. Details of frequency encoding/decoding of RS codes over $GF(2^m)$ are given in section D.1 of Appendix D.

6.2.1 Frequency domain decoder for RS codes

A time-domain encoder and frequency-domain decoder for RS codes is shown in Fig. 6.2. In Fig. 6.2 a Fourier transform is placed at the front end of the decoder and an inverse Fourier transform is needed at the end to provide the time domain decoder.

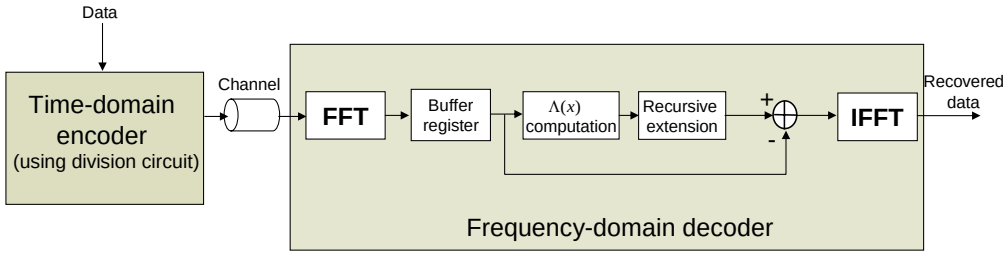

Figure 6.2: A time-domain encoder and frequency-domain decoder for RS codes

We note here that the decoder can be used with an encoder in the time domain, as shown in Fig. 6.2, or an encoder in the frequency domain, as shown in Fig. 6.3. If a time domain encoder is used, then the inverse Fourier transform of the corrected codeword spectrum $\mathbf{C}$ must be computed to obtain the time-domain codeword $\mathbf{c}$, from which the data is recovered. If instead, the encoder uses the data symbols in the frequency domain to specify the values of the spectrum, then the corrected spectrum gives the data symbols directly. That decoder does not compute an inverse Fourier transform. The decoder shown in Fig. D.1 has a simple structure for $k > 2t$ but when a pipelined decoder is desired, a more attractive approach can be used. A comparison between time and frequency domain decoding of RS codes can be found in section D.1.3 of Appendix D.

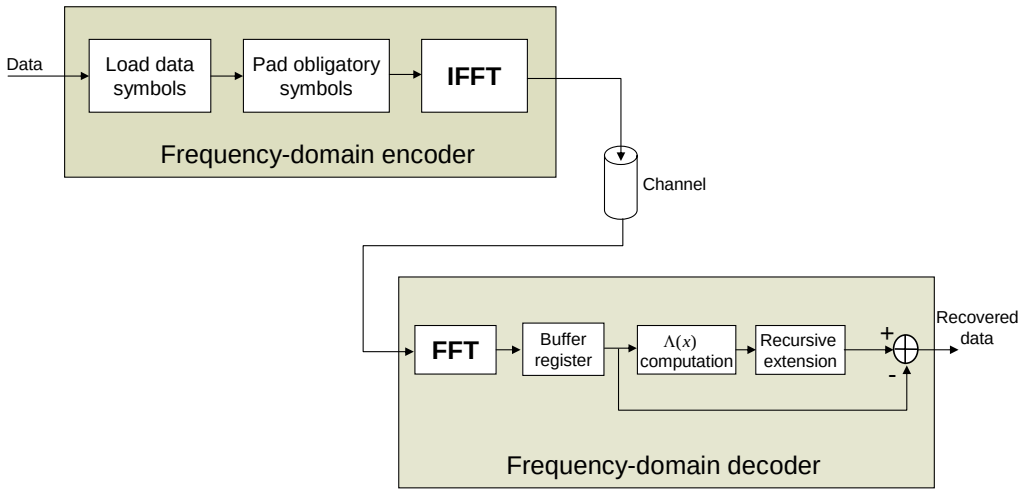

Figure 6.3: A frequency-domain encoder and frequency-domain decoder for RS codes

Frequency processing can play an important and central role in the encoding and decoding processes of RS codes. Compared to the time domain processing, the frequency domain processing presents important advantages in terms of running time and efficiency of hardware and software implementation of principal algorithms. But, the transforms used in

that frequency domain processing are defined over $GF(2^m)$ and they have the following two principal characteristics:

1. The transform length is equal to $2^m - 1$, where m is an integer.
2. The arithmetic operations are done modulo 2.

6.2.2 Fermat number transforms

Our aim is to insert the complex FFT operator in the channel coding function. However, the first characteristic which is the transform length of the finite field transform over $GF(2^m)$ does not match the one of the complex FFT defined over the complex field $\mathbb{C}$, which is of the form 2^m. This characteristic is a strong constraint that challenges the adaptation or the combination of the $GF(2^m)$ FFT structure with the complex FFT one, since the most efficient algorithms for the FFT computations are applied to transforms of length 2^m. Under this strong constraint, we are to find out a transform satisfying the complex FFT criteria while keeping in mind to include the classical finite field transform, used in the RS coding defined over $GF(2^m)$, in the intended common and reconfigurable FFT structure. Next section is dedicated to find the appropriate transforms. The strong constraint discussed above, has led us to seek other class of transforms which in turn will lead to other and specific class of RS codes. Our research on the state of the art of finite field transform and RS codes led us to spot specific class of transforms and get out the corresponding class of RS codes. These specific finite field transforms as well as the corresponding RS codes are defined over $GF(F_t)$, where

$$F_t = 2^{2^t} + 1, \tag{6.1}$$

and F_t is called the t^{th} Fermat number. These transforms called *Fermat Number Transforms*(FNT) constitute a subclass of the *Number Theoretic Transforms* (NTT).

The definition of NTT appeared in 1971 when Pollard [187], discussed transforms having the cyclic convolution property in a finite ring of integers. Agarwal [188], extended Rader's [189] Fermat Number transform theory by lengthening the transform size and showed that the usual FFT algorithm can be used to calculate the Fermat transforms. Justesen [190], proposed that transforms over the finite field $GF(F_t)$ can be used to define RS codes and improve the decoding efficiency of these codes. The treatment of RS codes over $GF(F_t)$ in frequency domain will constitute the window through which we look at the channel coding as being one of the communication tasks that can be performed using the FFT operator in the Software Radio context. Details of Fermat transform based RS codes is given in section D.2 of Appendix D. However we will show here the efficient role of the FNT in performing the most time-consuming steps of the decoding process. To better illustrate the task, we divide the decoding process into three phases as illustrated in Fig. 6.4. The first step consists of a Fermat transform to perform the syndrome computation, and the second performs the spectral analysis using the Berlekamp-Massey algorithm and some polynomial computations. The third step also consists of a Fermat transform to perform the Chien search followed by Forney algorithm to correct the erroneous symbols. As illustrated in Fig. 6.4, each of the syndrome computation and Chien search takes n clock cycles, where n is the block length of the code, which is the most long time in the

decoding process. The suggested FNT to be implemented to replace the corresponding circuit of each of the syndrome computation and Chien search has a transform length equal to power of 2. Thus, using the fast algorithms to compute the transforms, FNT can be implemented into $log_2 n$ stages where each stage is composed of $n/2$ computational units. By computational unit, we mean the *butterfly* consisting of a multiplier, adder and subtracter. This efficient implementation of the FNT permits to reduce each n cycles to $log_2 n$ cycles. A detailed performance comparison between RS over $GF(F_t)$ and RS over $GF(2^m)$ is presented in section D.2.3 of Appendix D.

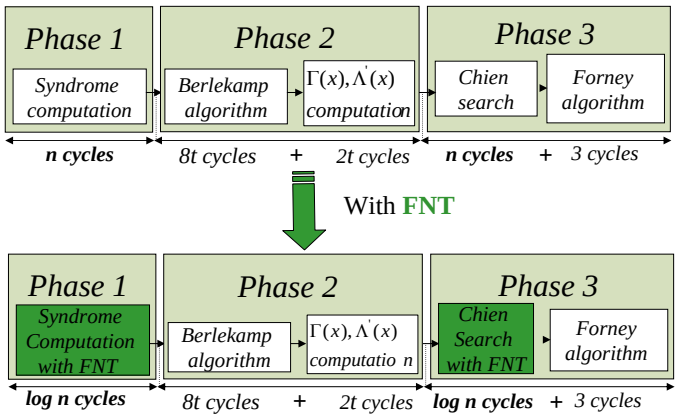

Figure 6.4: The three phases of RS decoding process over $GF(F_t)$

6.3 Dual mode FFT operator

In the previous section we described the efficient use of the FNT in the decoding process of the RS codes defined over $GF(F_t)$ and we discussed the possibility for designing it onto the FFT architecture. Now we turn our attention to the implementation of the global *dual mode FFT (DMFFT)* operator. We present the design and FPGA implementation of a reconfigurable DMFFT operator. In the present work, the target hardware for the implementation and evaluation of the proposed architectures is the Altera's FPGA devices using the traditional hardware description language VHDL. The design is based on the realization of reconfigurable arithmetical operators (multiplier, adder and subtracter) leading to *Reconfigurable Butterfly Processing Element* (RBPE) and ultimately DMFFT operator. This operator provides two functionalities: FFT and FNT. We can reconfigure DMFFT operator to perform either of the transforms. In case of Velcro operator we have FFT and FNT both, but each of the transform is implemented independently and due to this reason we call this operator as *Velcro* operator. Many different ways of implementing DMFFT operator are presented in section D.3 of Appendix D.

DMFFT Complexity Study

To evaluate the performances, the DMFFT operator was implemented on Stratix II and it was compared to *Velcro* FFT/FNT operator implemented on the same device.

Table 6.1: Comparison between the DMFFT-64 and the Velcro FFT/FNT-64 operator on a Stratix II, EP2S15F484C3

n_c	9	10	12	14	16
Velcro operator	4205 ALUTs 4.86 ns	4768 ALUTs 5.27 ns	5831 ALUTs 5.46 ns	6844 ALUTs 5.81 ns	8143 ALUTs 6.62 ns
DMFFT	3109 ALUTs 4.78 ns	3744 ALUTs 4.97 ns	4857 ALUTs 5.45 ns	5975 ALUTs 5.85 ns	7387 ALUTs 6.65 ns
Memory saving	33 %	31 %	27.2 %	24.3 %	21.9 %
ALUT's gain	26 %	21.4 %	16.7 %	12.7 %	9.2 %
η's gain	37.4 %	35 %	20 %	13.9 %	9.7 %

Table 6.1 shows the implementation measures for the DMFFT-64 implemented for different n_c i.e. word-lengths in $\mathbb{C}$. The Fourier/Fermat transforms that can be performed in this same architecture have $N = 64$ as transform length. Regarding the Fourier transforms, the wordlength plays an important role on the accuracy of the computations. This accuracy increases as the wordlength increases. As for the Fermat transforms, they are defined over $GF(257)$ whose symbols are represented by 9 bits. The complexity study for Velcro FFT and FNT-64 operators, given in Table 6.1 was drawn by evaluating their performance-to-cost ratio defined in [44]. The ratio η is expressed as $\frac{1}{TC} * 10^6$, where C is the number of logic blocks related to the cost of a FPGA-based circuit and T the execution time in nanoseconds (ns). As η increases, a better trade-off between execution time and cost is obtained. The implementation results showed, in favor of the DMFFT operator, an important memory saving and gains in terms of ALUTs and in terms of performance-to-cost ratio which vary with the wordlength and the transform length.

Table 6.2 represents the implementation measures of the DMFFT-256 and the Velcro FFT/FNT-256 operator. The memory saving is practically the same and the other gains evolve in the same manner of that of DMFFT-64 but with lower values. This is due to the fact that the DMFFT complexity is dominated by that of the FFT and when the transform length increases, the FFT complexity increases rapidly while the FNT complexity remains moderate.

This illustrates the interest to use common operators' solutions rather than Velcro in the design, in terms of complexity of implementation. Another important advantage in soft-

Table 6.2: Comparison between the DMFFT-256 and the Velcro FFT/FNT-256 operator on a Stratix II, EP2S15F484C3

n_c	9	10	12	14	16
Velcro operator	5327 ALUTs 5.02 ns	6079 ALUTs 4.95 ns	7518 ALUTs 5.45 ns	8966 ALUTs 5.84 ns	10770 ALUTs 6.55 ns
DMFFT	4466 ALUTs 4.86 ns	5365 ALUTs 4.90 ns	6819 ALUTs 5.50 ns	8564 ALUTs 5.90 ns	10389 ALUTs 6.58 ns
Memory saving	33 %	31 %	27 %	24 %	21.9 %
ALUT's gain	16.2 %	11.7%	9.29 %	4.68 %	3.5%
η's gain	24 %	15.1 %	9.4%	4.2 %	3.54 %

ware radio is that in the case of a *Velcro*, the reconfiguration time is more disadvantageous than a change of parameters of the DMFFT i.e. parametrisation allows us to minimize reconfiguration time. Apart from a CR vertical handover scheme, it does not matter in the case of inter-standards' use of the same operator as we make the assumption of a sequential use of standards with a time between two unconstrained communications. But if the operator is split among several features of a single standard, this is crucial since it will perform at a very high rate of change of functionality of the operator.

In a nutshell the built DMFFT operator is a potential candidate for a common operator, to be integrated in an SDR system intending to support several standards where the complex Fourier transform and the RS coding are required. The RS codes that can be teated with this operator are the RS codes defined over $GF(F_t)$. In the next section we describe our proposed methodology for the design of multi-standard systems based on idea of common operators.

6.4 Application

In this section we present a design scenario that explains our approach to design multi-standards systems. Let us first summarize our discussion. So far we have come to the conclusion that our DMFFT operator can be used to compute FFT and FNT and that if we use RS defined over $GF(F_t)$ instead of $GF(2^m)$ then we have the advantage of using the same DMFFT to compute some steps of RS as well. As a design scenario we select three air interface standards namely WiFi, WiMAX and xDSL. We are well aware of the fact that the WiMAX and xDSL standards have already been established with $GF(2^m)$ RS coding scheme. We argue here that with the proposed RS coding scheme over $GF(F_t)$ in this chapter, some complexity could have been saved because two computationally intensive steps of RS decoding (syndrome calculations and Chien search) can be performed with the DMFFT. As mentioned earlier this DMFFT operator can be a good candidate

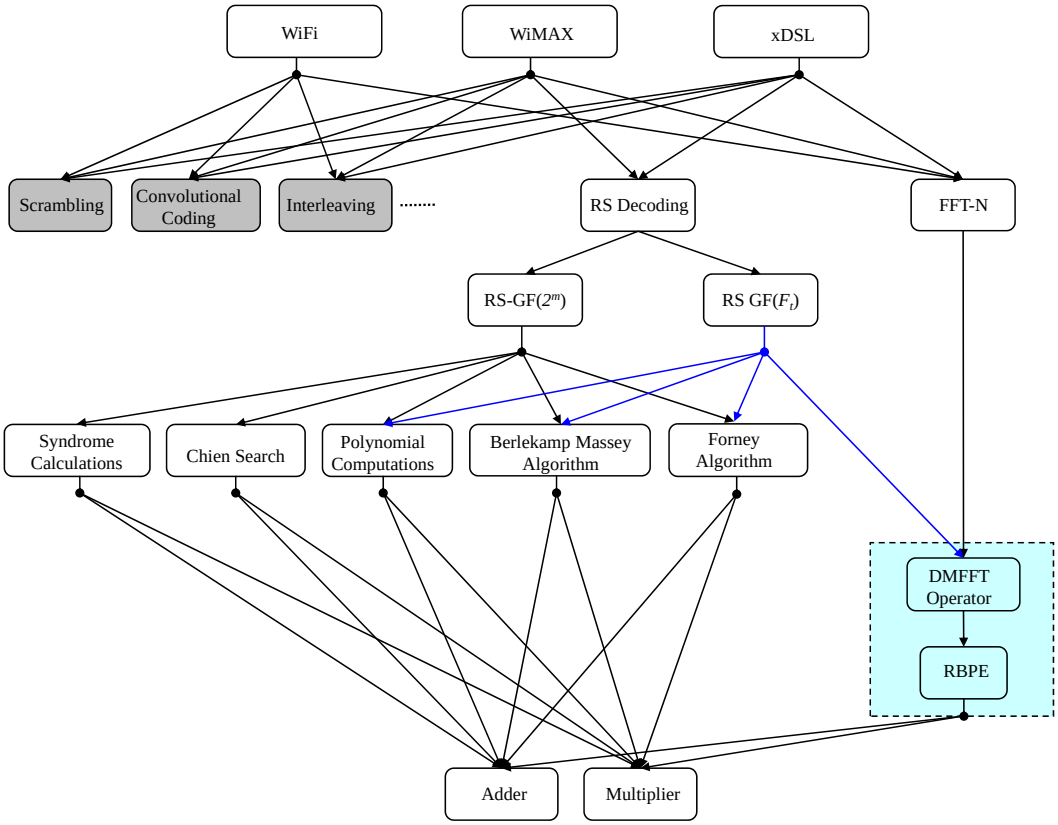

Figure 6.5: Graph of tri-standard system if RS coding in $GF(F_t)$ had been used

for common operator. If we consider DMFFT as one of the common operators and LFSR as the other common operator then the whole system can be implemented by these two operators. Complete details of the procedure for selection of these operators can be found in part II of this thesis, as mentioned earlier.

Let us consider the graph of tri-standard system as shown in the Fig. 6.5. On the top of this figure are the standards themselves and as we move from top to bottom we pass through various granularity levels and finally reach at the bottom level that consists of adder and multiplier. The boxes shown in the gray color are not to be considered in this chapter because they are implemented with LFSR common operator as mentioned in next chapter.

In this figure we concentrate only on the Reed Solomon decoding and FFT-N blocks. All of the three standards require FFT of different size and in addition WiMAX and xDSL require the RS. For the RS we have two options, either we can use a classical RS decoding scheme based on $GF(2^m)$ or we can use the proposed RS decoding scheme based on $GF(F_t)$. It is to be noted here that if we may have selected this new type of RS coding/decoding scheme in WiFi, WiMAX and xDSL then;

1. We would have obtained the same performance results as mentioned in section D.2.3.
2. With reduced complexity as given in tables 6.1 and 6.2.

Let us design this tri-standard systems with minimal cost. Instead of considering the whole system we concentrate first on the most computational intensive part of the system i.e. the computations of FFT. Then we integrate in the graph, the steps of the RS that can be performed by the DMFFT i.e syndrome computations and Chien search. As shown in the Fig. 6.5 we have two choices to do the RS decoding; either we can use the classical one or we can take the advantage of using the DMFFT. Zooming the dotted portion from the complete graph results is the Fig. 6.6. As visible in this Figure we have some parameters associated with each arrow and with each block. These parameters are not shown in the complete graph for the purpose of clarity.

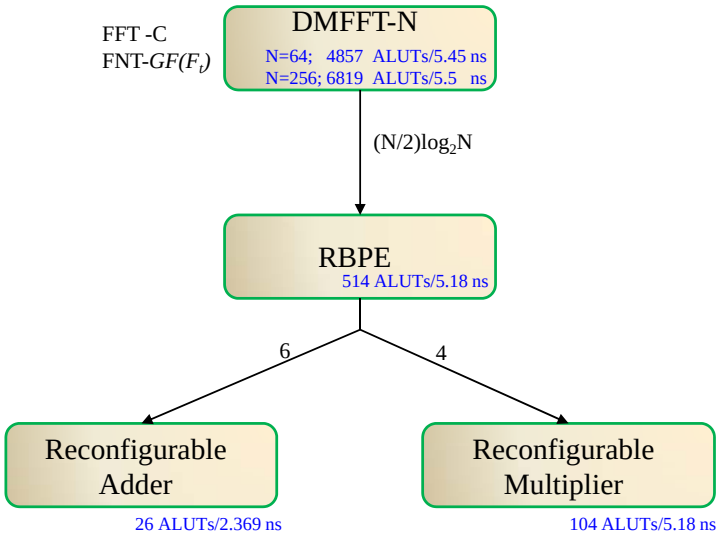

Figure 6.6: Building block of DMFFT

The parameter associated with the arrows corresponds to the number of calls. These are the number of times a block at level *n-1* will be invoked to perform the task of the block at level *n*. This means that same task can be performed by a block at level *n* instead of block at level *n-1*, in less time but with much higher cost and vice versa as explained in part II. It is to be noted that certain weights that are associated with arcs are parameterizable e.g. the weight associated with OR hyperarc between DMFFT-N and RBPE as shown in Fig. 6.6. Suppose we are to perform FFT-64. Now this FFT-64 can be performed by DMFFT-64 or we can do the same with the RBPE. The only difference will be that the DMFFT-64 will be called once whereas RBPE will be called $(N/2)$ times to perform FFT-64. This is because of the fact that for FFT-N we have chosen to implement 1 butterfly per stage, where total number of stages are $log_2 N$. Hence in our design scheme we already have $log_2 N$ butterflies that are to be called (N/2) times to perform FFT-N.

The parameter associated with the block corresponds to its cost. By cost of the block we mean its installation cost and it can be the implementation cost of the block in FPGA in terms of ALUT. Here in this figure the cost mentioned is in terms if ALUTs. These costs for the DMFFT and lower blocks are listed in Table 6.3.

Table 6.3: No. of call of individual blocks of DMFFT

Block Name	**No of ALUT**	**Execution Time(ns)**	**No of Calls**
DMFFT-256	6819	5.5	1
RBPE	514	5.18	128= $(N/2)$
Re-Adder	26	2.37	6
Re-Multiplier	104	5.18	4

Table 6.4: No. of calls of multiplications and additions for classical RS decoding for t=8 and N=256

No. of Calls	**Additions**	**Multiplications**
Syndrome Comp.	4096	4096
Chien Search	2048	2048
Polynomial Comp.	384	392
Berlekamp Massey	512	512
Forney Algorithm	8	16

As an example of cost computation, let us consider the RS decoding node in the graph. Complete procedure for cost computations has already been explained in chapter 5. Table 6.4 list the number of calls for additions and multiplications for different functions (classical RS decoding), e.g. for syndrome calculations, an adder will be called 4096 time and multiplier will be called 4096 times. For the RS decoding node we have two options to find the costs: either by classical RS or by RS defined over $GF(F_t)$. Our aim here is just to further illustrate the cost computations.

The cost of the RS decoding node can be computed by equation (4.10) duplicated below:

$$Cost_{level} = \Big(\sum_{i=1}^{i=m} CC_i NoC_i \Big) \times NoC_{i+1} + \sum_{i=1}^{i=m} BC_i.N_i \tag{6.2}$$

$N_i = 1$ since cost involves only those nodes that are present in the system. As we are considering RS decoding node only, hence $NoC_{i+1} = 1$ and, equation (6.2) becomes:

$$Cost_{RSDecoding} = \Big(\sum_{i=1}^{i=5} CC_i NoC_i \Big) \times NoC_{i+1} + \sum_{i=1}^{i=5} BC_i \tag{6.3}$$

Using the first option to perform RS decoding, equation (6.3) becomes:

$$\begin{aligned} Cost_{RS-GF(2^m)} &= Cost_{Syndrome} + Cost_{Chien} + Cost_{Poly} \\ &+ Cost_{Berlekamp} + Cost_{Forney} \end{aligned} \tag{6.4}$$

or,

$$\begin{aligned} Cost_{RS-GF(2^m)} &= \big((4096 + 2048 + 392 + 512 + 16) \times CC_{Multiplier} \\ &+ (4096 + 2048 + 384 + 512 + 8) \times CC_{Adder}\big) \times 1 \\ &+ BC_{Multiplier} + BC_{Adder} \end{aligned} \tag{6.5}$$

or,

$$Cost_{RS-GF(2^m)} = \big(7064 \times 5.18 + 7048 \times 2.37\big) \times 1 + 104 + 26 = 53425.28 \tag{6.6}$$

Using the second option to perform the RS decoding, equation (6.3) becomes:

$$Cost_{RS-GF(F_t)} = Cost_{Poly} + Cost_{Berlekamp} + Cost_{Forney} + Cost_{DMFFT} \tag{6.7}$$

or,

$$\begin{aligned} Cost_{RS-GF(F_t)} &= \big((392 + 512 + 16) \times CC_{Multiplier} \\ &+ (384 + 512 + 8) \times CC_{Adder} + 2 \times CC_{DMFFT}\big) \times 1 \\ &+ BC_{Multiplier} + BC_{Adder} + BC_{DMFFT} \end{aligned} \tag{6.8}$$

or,

$$Cost_{RS-GF(F_t)} = \big(920 \times 5.18 + 904 \times 2.37 + 2 \times 5.5\big) \times 1 + 104 + 26 + 6819 = 13868.08 \tag{6.9}$$

Considering the RS decoding node only we have the following cost function:

$$\mathbf{C_{RSDecoding}} = \min\big(Cost_{RS-GF(2^m)}, Cost_{RS-GF(F_t)}\big) = \mathbf{Cost_{RS-GF(F_t)}} = 13868.08$$

Results show that the cost is much less in case of RS decoding in GF(F_t). This illustrates the interest of using DMFFT as a common operator.
Let us further explore the costs of this DMFFT operator. If the DMFFT is considered only then the cost will be as follows:

$$\begin{aligned} Cost_{DMFFT} &= BC_{DMFFT} + CC_{DMFFT} \\ &= 6819 + 5.5 = 6824.5 \end{aligned} \tag{6.10}$$

As DMFFT operator can be implemented by RBPE. If we use RBPE for its implementation the cost of DMFFT (using equation (4.10)) will be the following:

$$\begin{aligned} Cost_{DMFFT_{RBPE}} &= CC_{RBPE} \times NoC + BC_{RBPE} \\ &= 5.18 \times 128 + 514 \\ &= 1177.04 \end{aligned} \tag{6.11}$$

By going down in the graph the RBPE can be implemented by adders and multiplier. Using adders and multiplier for implementation, the cost of DMFFT using equation (4.10)) can be found as follows:

$$Cost_{DMFFT_{Adder+Multiplier}} = \Big(\sum_{i=1}^{i=2} CC_i NoC_i\Big) \times NoC_{i+1} + \sum_{i=1}^{i=2} BC_i \tag{6.12}$$

or,

$$\begin{aligned} Cost_{DMFFT_{Adder+Multiplier}} &= \Big(CC_{Adder} \times NoC_{Adder} + \\ &\quad CC_{Multiplier} \times NoC_{Multiplier}\Big) \times 128 + \\ &\quad BC_{Adder} + BC_{Multiplier} \end{aligned} \tag{6.13}$$

(6.14)

or,

$$Cost_{DMFFT_{Adder+Multiplier}} = \Big(2.37 \times 6 + 5.18 \times 4\Big) \times 128 + 26 + 104 = 4601.55 \quad (6.15)$$

$$(6.16)$$

These results show that implementation of DMFFT by RBPE gives minimum cost as:

$$Cost_{DMFFT_{RPBE}} < Cost_{DMFFT_{Adder+Multiplier}} < Cost_{DMFFT} \quad (6.17)$$

In other words result of equation (6.9) using RBPE for DMFFT's implementation will now become:

$$Cost_{RS-GF(F_t)} = 8878.16 \quad (6.18)$$

or,

$$\mathbf{C_{RSDecoding}} = \mathbf{Cost_{RS-GF(F_t)}} = 8878.16 \quad (6.19)$$

This indicate that implementation of DMFFT through RBPE results in a further reduction in cost.

Let us discuss the trade-off between BC and CC. Considering equation 6.10 (in which we consider the cost at level of DMFFT node) we can see that compared to BC, the CC is negligible. Now if we consider the equation 6.15, we can see that compared to CC, the BC is negligible. These are two extreme levels for finding the cost of DMFFT. At highest level (6.10) BC is higher but CC is lower and at lowest level (6.15) BC is lower and CC is higher. Considering the cost of DMFFT using an intermediate level of RBPE in equation 6.11, we see that BC and CC are comparable. RBPE is at the best level of granularity that we are seeking for. This is exactly in accordance with our aim of finding a balance between complexity and computational efficiency.

This optimization problem can be solved by simulated annealing as explained in part II of this thesis. But for very small portions of the graph we can use exhaustive search to solve the optimization problem. After substituting the values of parameters for arrows and blocks as given in Table 6.4, we run the optimization algorithm. Result of optimization algorithm choose RBPE to be used as common operator as it gives the solution with minimal cost. It is fair enough to use exhaustive search here for finding the optimized solution instead of using the simulated annealing as the part of the graph which is being addressed is quite small. Of course for the complete graph we do not have the choice of exhaustive search and we have to use simulated annealing.

6.5 Conclusions

In this chapter we addressed the problem of parametrisation using the common operators' technique, where it is in line with the optimal design of a SDR system intended to support several communication standards.

In this context, we investigated the frequency processing of RS codes with the aim to insert the classical FFT operator, initially used for complex Fourier transform, in the encoding and decoding processes of RS codes. To be able to exploit the whole FFT-$\mathbb{C}$ structure we explored finite field transforms having a highly composite length. The candidate transforms were the Fermat transforms defined over $GF(F_t)$. These transforms led us to revive a specific class of RS codes defined over $GF(F_t)$.

The redesign of the FFT-$\mathbb{C}$ was explained in such a way to be able to provide two functionalities: complex Fourier transform and Fermat transform [19, 45]. Conceptually, the design of such a dual mode operator implies the design of arithmetical operators capable to operate over the two domains: $\mathbb{C}$ and $GF(F_t)$. We proposed a reconfigurable architecture for the butterfly that constitutes the core of the dual mode operator. Based on the FFT-$\mathbb{C}$ structural strategy, we have designed the architecture of the DMFFT operator.

To evaluate the complexity and speed performances of this operator, we have considered its implementation on Altera's FPGA devices. Compared to a Velcro FFT/FNT operator, the DMFFT presented an important gain in terms of ALUTs and memory saving.
We illustrated our design approach through a simplified sub-design. Results of running the optimization algorithms on subpart of the graph show that RBPE offers minimal cost when designing of multi-standard systems is considered i.e. RBPE is selected as CO. Hence the implementation results are being validated by our tool. These results have been accepted for a joint publication in [45].

Moreover, in order to show the scope of the proposed design method, we also demonstrated in this study that if the CO approach for reconfigurability and implementation complexity would have been considered like this (the context of possible multi-standards equipments) in WiFi, WiMAX, xDSL standardization, then RS decoding in $GF(F_t)$ could have been chosen.

In the next chapter we present the use of LFSR as common operators in the design of multi-standards SDR systems.

Chapter 7

Case Study: LFSRs as Common Operators in multi-standards SDR systems

Contents

7.1 Introduction

In the previous chapter we described the DMFFT as a common operator. In this chapter we describe the *linear feedback shift register* (LFSR) as a *common operator* (CO) which is one of the first implementation of common operator technique on the area of techniques of parameterizations. We explain our four developed common architectures to design 16 LFSR COs that can carry out several operations of multi-standard SDR/CR receivers. We also give the implementation details on Altera Cyclone II. Either a single or combination of these LFSR operators can be used to perform the functions as necessitated by the different air interfaces to be supported by SDR.

We start this chapter with a brief introduction to LFSR. A detailed introduction to basics of LFSRs and the architectures of different LFSR operators are given in section E.1 of

Appendix E. In section E.1 of Appendix E, we start our discussion with classical LFSR and then identify the operations that can be performed with LFSRs. Based on two basic families of LFSRs i.e. Fibonacci and Galois we explain the structure of *Reconfigurable Fibonacci LFSR* (RF-LFSR) and *Reconfigurable Galois LFSR* (RG-LFSR) operators [46]. Extending the functionality of these basic LFSR operators we explain the architectures of developed *R-LFSR* and *ER-LFSR* which are generic in nature and support more LFSR based operations than the basic LFSR operators. These architectures are dimensioned to obtain two common operators, namely x-LSFR4 and x-LFSR8. We give the implementation details of these operators in section 7.3. We present a design scenario in section 7.4 which consists of designing the 3GPP LTE (Two modes) and WiFi (Single mode) tri-standard transceiver. For the design we have choice between different LFSR operators. We explain the decomposition of different LFSR operators. By using procedures detailed in part II of this thesis, we determine which combinations of LFSR are best suited for performing multi-functions of single standard and multi-functions of multi-standards. Finally, results and conclusions' section ends this chapter.

7.2 Linear feedback shift register

A linear feedback shift register is a shift register whose input bit is a linear function of its registers' previous contents. The operation of the register is deterministic and the sequence of values produced by the register is completely determined by its current (or previous) state. Two families of LFSR are clearly discriminated in the literature [119]:

- Fibonacci LFSRs and
- Galois LFSRs

These two types are able to carry out dissimilar functionalities and as a consequence, we design two distinct COs, called *Reconfigurable Fibonacci LFSR* (RF-LFSR)and *Reconfigurable Galois LFSR* (RG-LFSR). In order to enlarge the workable operations of RF/RG-LFSRs, we undergo the second step of the pragmatic approach as explained in part I of this thesis and design step by step two new common operators; the *Reconfigurable LFSR* (R-LFSR) and the *Extended Reconfigurable LFSR* (ER-LFSR). In the next section we explain the implementation of LFSR COs.

7.3 Implementation of common operators

7.3.1 Classical approach

Once, the COs designed, we must find an implementation to carry out all the structures to replace. The classical approach to execute a CO is to schedule the operator, i.e. implement the minimum number of entities required and allocate the different *calls* on the so-implemented operators as explained in part II. LFSR architectures should be considered differently. Actually, for each output, the LFSR process needs not only the input but also the value of all the registers. As a result to schedule a LFSR operator and use it several times for successive operations, we have no choice but to store all the content of the registers, the parameters of the loops, the inputs and outputs. Moreover, even if the

structures are replaceable, say by the ER-LFSR, they exhibit different configurations. For instance, in this chapter, the CO required must deal with one LFSR chain of 32 registers, two chains of 27 registers or three chains of 8 registers. From this point, the design and the set-up of one specific Common Operator based on ER-LFSR architectures is a main issue.

7.3.2 Proposed solution: common operator bank

To overcome all the drawbacks inferred by scheduling, we decide to obscure the scheduling issue. Following the design of filter banks, we create a Common Operator Bank (COB), where small ER-LFSR operators (Constituent Component) of identical sizes are interconnected. These connections are parameterizable and allow us to implement at the same time ER-LFSR chains of different sizes. The purpose of this bank is to avoid the scheduling, by settling down as many operators as required by the most demanding standard. Taking the worst case may seem in contradiction to CO philosophy because CO technique tends to find the smallest common entity (in fact it is the less expensive entity). Moreover, worst case approach may seem obvious but it offers an economic advantage compared to Velcro which is coherent with CO's goals. However, we will see later that it is the CO able to run the worst case (which means automatically all the other cases) which are necessary in the equipment will be chosen. The operator bank was created to replace all the operations enforceable at the same time by one standard. Fig. 7.1 illustrates different possible allocations. With the COB, we replace *exactly* the structure to address, without the need to add extra mapping. With respect to the smallest and the most restated structures to implement, we use constituent components of size 4 and 8 which will be explained in the beginning of next section.

7.3.3 Use of common operator banks

Let us describe the implementation of operators on Field Programmable Gated Array (FPGA). In the implementation on Altera Cyclone II the basic element is *logic cell*. Number of logic cells required to implement a single structure are given in Table 7.1. Minimal number of constituent components required to implement various operators for different functions are listed in Table 7.2. Using Table 7.1 and 7.2 the number of logic cells required can be easily found.
In these Tables RG-LFSR4/8 means that an LFSR CO is constituted of 4/8 RG-LFSR units. This is also the size of the CO. As mentioned earlier, we have chosen the size of 4 and 8 only. Let us explain the selection of size 4 and 8. Before selecting these sizes, different LFSR structures with sizes starting from $2, 4, 5 \ldots 30$ were evaluated with respect to their complexity in terms of transistors. LFSRs of size 4 gives smallest complexity than all other sizes. In addition, size 4 is the smallest size required by any structure to be replaced. Size 8 is the most used size in the structures to be replaced because of multiple of 8. So we consider only LFSR-4 or LFSR-8 whatever the type of LFSR may be i.e RG/RF/R/ER-LFSR.
Let us explain the figures given in Table 7.2. This table gives the number of LFSRs required to implement different functions of different standards. We first consider the case of single standard only (say WiFi) and explain the table for WiFi. In WiFi, three functions, namely scrambling, CRC and convolutional coding (CC) can be performed by the

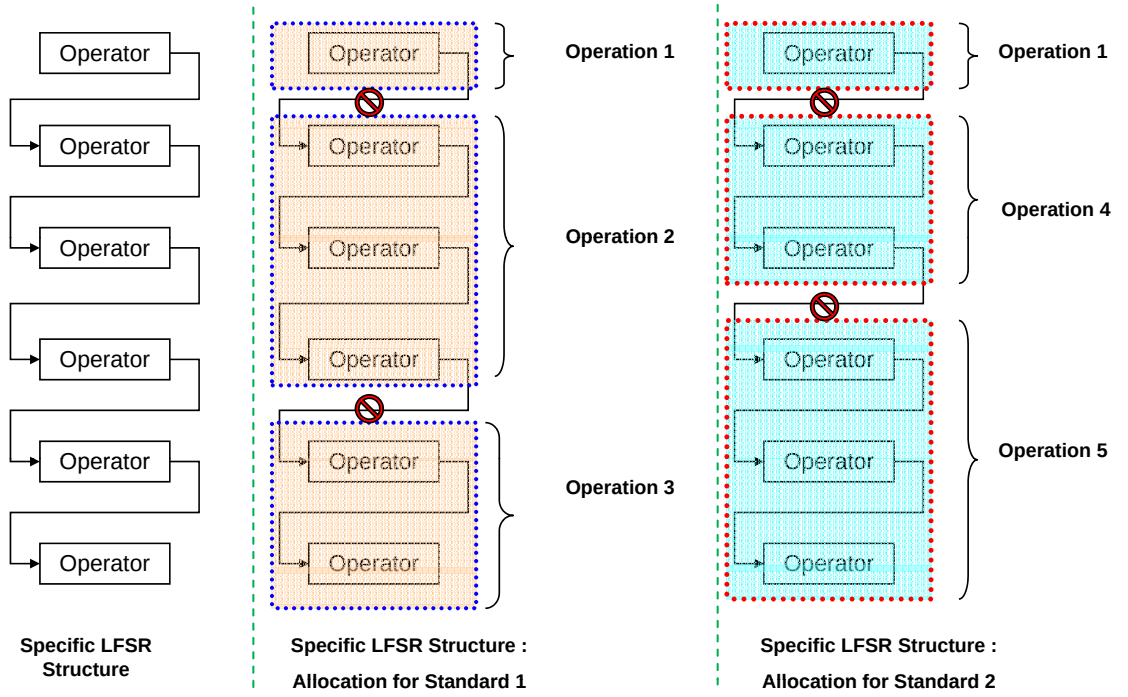

Figure 7.1: Common operator bank

Table 7.1: Independent LFSR decomposition

LFSR common operators	**Logic cells**
RG-LFSR4	5
RG-LFSR8	10
RF-LFSR4	5
RF-LFSR8	10
R-LFSR4	6
R-LFSR8	10
ER-LFSR4	11
ER-LFSR8	16

use of LFSRs. As far as LFSRs are concerned we have eight different LFSR operators to implement these functions listed on top row of the Table 7.2. From here onwards in the text in this chapter, operators means LFSR operators.

To implement scrambling we can use either of these eight operators. Now if we want to implement scrambling and CRC at the same time we note that RF-LFSR4 and RF-LFSR8 cannot be used to implement CRC and for CRC we have a choice between remaining six operators. Either we can use any of remaining six operators to implement both scrambling and CRC e.g. RG-LFSR4 (4×RG-LFSR4 for scrambling + 8 ×RG-LFSR4 for CRC) or we can use a combination of different LFSR operators e.g. we may implement scrambling by RF-LFSR4 (4×RF-LFSR4 for scrambling) and CRC by RG-LFSR8 (4×RG-LFSR8 for

CRC).

Further if we want to implement scrambling, CRC and CC (IEEE 802.11b) we cannot use RF-LFSR4/8 and RG-LFSR4/8 for CC and now we have choice between remaining four operators for CC. As discussed above a single operator or a combination of different operators can be used to implement these three functions. If we want to implement all the functions of WiFi (based on LFSR) we note that for CC's (IEEE 802.11g) we are left with no choice but ER-LFSR4/8. Again we can use single operator or a combination of operators to implement all the four functions for WiFi. Let us assume that we want to implement both CC schemes for WiFi. In this case we have to use either ER-LFSR4 or ER-LFSR8. Assume that we choose ER-LFSR4 and now in this case we take $max(6, 4) = 6$ ER-LFSR4 operators to implement both CC schemes. It is important to note that with six ER-LFSR4 operators we are able to implement the more complex of the two schemes and hence less complex will be automatically taken into account by the complex one so we need only 6 ER-LFSR4 operators and not $6 + 4 = 10$ ER-LFSR operators.

Let us consider the case of multi-standards systems. Suppose that we want to support the standards WiFi, WiMAX and 3GPP LTE and we want to implement CRC for all the three standards. For WiFi we need to support CRC-16 and it can be implemented by either of 6 operators. For WiMAX we need to support CRC-32 and CRC-16 and these can be implemented by either of six operators. For 3GPP LTE we need to support CRC-24/16/12/8 and we have choice between six operators. Let us suppose that we select R-LFSR4 operator to implement CRC for the three standards. So for WiFi we need 8 R-LFSR4 operators, for WiMAX we need $max(16, 8) = 16$ R-LFSR4 operators and for 3GPP LTE we need $max(12, 8, 6, 4) = 12$ R-LFSR4 operators. Considering all the three systems to implement CRC for all by R-LFSR4 we need $max(8, 16, 12) = 16$ R-LFSR4 operators. As mentioned earlier, if the more complex is implemented the less complex will be automatically taken care of. Same principle applies for the implementation of other functions mentioned in Table 7.2.

7.3.4 Implementation issues

From this previous explanation, it is appropriate to say that many implementation are possible depending upon the standard to be implemented. We may decide on one homogeneous COB with only one type of constituent component or we can opt for a combination of distinct constituent components and design a heterogeneous COB. The choice is versatile. An implementation for each possibility is inconceivable. All this previous work needs a specific mean to point out the best solution to implement and the present optimization process comes up to that expectation.

A first attempt on a tri-standard terminal considering two LTE modes and one WIFI mode revealed that the implementation of the COB designed with ER-LFSR8 as a constituent component required almost 5% of Logic Cells more than the equivalent Velcro Method. In fact these are the COB designed with ER-LFSR8 that use 352 Logic Cells as compared to 336 Logic Cells used by Velcro method (results obtained by synthesis). The initial will of the CO is to make implementation better. As result in the next section, we will apply

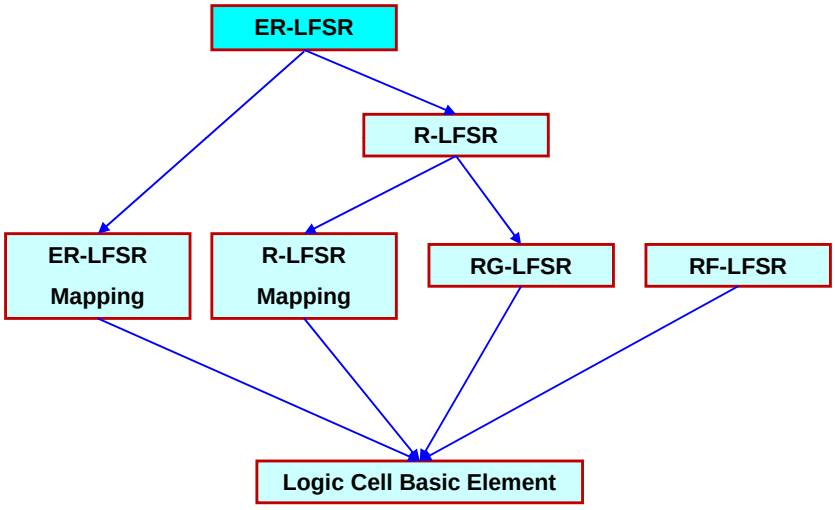

Figure 7.2: Combined LFSRs

the optimization process on this example to identify the best reachable solution, using all the possible size and type of COB.

7.4 Application

7.4.1 Integration of LFSR in the graphical approach

In this section we consider the different LFSRs as the evolution of other LFSRs of *smaller* size and as a consequence, we modify the design of the different operators to evolve in a more complex LFSR with a simple *add-on*. In this new design both functional perimeters and LFSR are *nested*. As explained in the previous section, the basic entity in the FPGA implementation is the Logic Cell. The combined LFSR approach is shown in Fig. 7.2. The results of this implementation are summed up in the Table 7.3.

With this kind of decomposition, we could see the (E)R-LFSR as an addition of R(G)-LFSR plus others structures based on Logic Elements:

- R-LFSR = RG-LFSR + 1.Logic Cells (R-LFSR Mapping)
- R-LFSR4 = RG-LFSR4 + 5.Logic Cells (R-LFSR4 Mapping)
- R-LFSR8 = RG-LFSR8 + 9.Logic Cells (R-LFSR8 Mapping)

and

- ER-LFSR = R-LFSR + 1.Logic Cells (ER-LFSR Mapping) = RG-LFSR + 1.Logic Cells
- ER-LFSR4 = R-LFSR4 + 6.Logic Cells (ER-LFSR4 Mapping)= RG-LFSR4 + 11.Logic Cells.

Table 7.2: Operations of WiFi/WiMAX/3GPP LTE and Number of LFSRs

Function	**RG-LFSR4**	**RG-LFSR8**	**RF-LFSR4**	**RF-LFSR8**	**R-LFSR4**	**R-LFSR8**	**ER-LFSR4**	**ER-LFSR8**
WiFi-Complexity in terms of number of LFSRs								
Scrambler	4	2	4	2	4	2	4	2
CRC16	-	-	8	4	8	4	8	4
CC-802.11b	-	-	-	-	4	2	4	2
CC-802.11g	-	-	-	-	-	-	6	6
IEEE 802.16 d/e (WiMAX) Complexity in terms of number of LFSRs								
Scrambler-15	8	4	8	4	8	4	8	4
+Scrm.-11	6	4	6	4	6	4	6	4
or +Scrm.-22	12	6	12	6	12	6	12	6
CRC32	-	-	16	8	16	8	16	8
CRC16	-	-	8	4	8	4	8	4
CC-1/2	-	-	-	-	4	2	4	2
TC-1/2	-	-	-	-	-	-	1	1
TC-1/3	-	-	-	-	-	-	2	2
3GPP LTE Complexity in terms of number of LFSRs								
Scrambler-UL	8	4	8	4	8	4	8	4
+Scrm.-DL	6	4	6	4	6	4	6	4
CRC32	-	-	12	6	12	6	12	6
CRC24	-	-	8	4	8	4	8	4
CRC16	-	-	6	4	6	4	6	4
CRC8	-	-	-	-	4	2	4	2
CC-1/2	-	-	-	-	4	2	4	2
CC-1/3	-	-	-	-	6	3	6	3
TC	-	-	-	-	2	2	2	2

Table 7.3: Combined LFSR decomposition

Combined LFSR common operators	Logic cells
RG-LFSR4	5
RG-LFSR8	10
RF-LFSR4	6
RF-LFSR8	11
R-LFSR4	11
R-LFSR8	20
ER-LFSR4	17
ER-LFSR8	27

- ER-LFSR8 = R-LFSR8 + 7.Logic Cells (ER-LFSR8 Mapping) = RG-LFSR8+ 16.Logic Cells.

It is to be noted that above relations are just to show the principle of combining the operators. These relations are not to be considered for calculating equivalent costs. As a consequence, every LFSR operator can have more than one costs depending upon the type of lower LFSR chosen to combine it into the higher one.
In the *combined operator's* approach, we create a first entity, in this case, the R-LFSR and we supplement this basic entity with other structures to obtain evolved operators such as R-LFSR and ER-LFSR. The difference with the *independent operator's* approach consists in the implementation of the R-LFSR and ER-LFSR. In the second one, they are directly implemented whereas in the first one, they are the result of the mapping and the addition of existing structures. Consequently, the combined operator approach is more demanding than the independent operator approach.

7.4.2 Design scenario

In this section we present a design scenario that explains our approach to design multi-standards systems. Let us first summarize our discussion. So far we have presented our LFSR operators that can be used to replace many functions of various air interface standards. We had the results of implementing these operators on FPGA and we wanted to verify these results by our graphical approach using the optimization algorithm. So our aim is to validate the results already obtained by our simulation tool. The results of implementation cannot be used directly into our graphical approach hence we made appropriate changes so that we can use these results as mentioned earlier.

As a design scenario we select two OFDM based air interface standards namely 3GPP LTE (Two modes) and WiFi (Single mode). We want to replace the functions of scrambling, CRC and convolutional coding (CC) by our operators. To replace these functions we have choice among different available LFSR operators. We can use either a single operator or a combination of multiple operators to perform the various operations depending upon designer's needs. Procedure for selection of these has already been addressed in part II of this thesis.

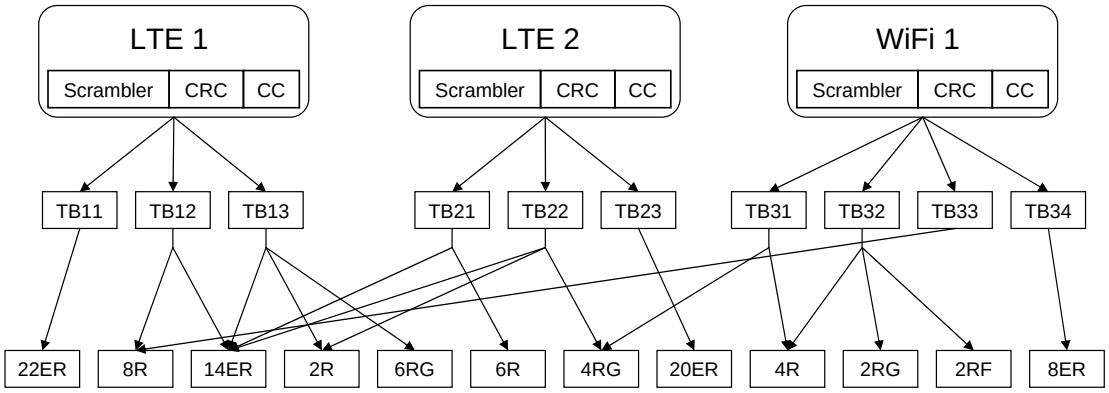

Figure 7.3: Implementation of a tri-standard systems with COs

Let us design this system with minimal cost. Instead of considering the whole system we concentrate only on the functions that can be performed by LFSRs. The graph of this system is shown in the Fig. 7.3. In Fig. 7.3 TBxx stands for transparent box, introduced to facilitate the grouping of different LFSR operators. At the bottom of this figure each LFSR operator is preceded by a number that indicates the number of duplications of that operator, e.g. 22ER means that one ER-LFSR4/8 operator is duplicated 22 times. Same is the case with other operators like 8R, 14ER etc.

On the top of this figure are the standards themselves i.e. LTE1, LTE2 and WiFi1. All the three standards require scrambling, convolutional coding and CRC to be performed. Lets consider LTE1 only. The three functions for the LTE1 can be performed by either a single CO i.e. 22ER or a combination of 8R and 14ER or a combination of 14ER, 2R and 6RG as shown in the left part of Fig. 7.3. LTE2 can use either a single CO or a combination of new and existing COs for LTE1. Same is the case with WiFi1. All the possibilities for implementing the three functions for three standards are shown in Fig. 7.3. Now, we want to find out the best combination of the operators that will give us the minimum number of operators with minimum cost. This should be noted that in this figure we have not considered the decomposition of bigger operators into smaller ones to keep the example fairly simple. This decomposition will be introduced in the second half of this example. To find the minimum cost we are to run the optimization algorithm.

For the optimization, we calculate the cost of each block and associate this cost with each block. Let us discuss the cost parameters here. We are to put BC/CC with each node/PE and NoC with arrows as mentioned earlier. The BC costs can be directly calculated from the number of LFSRs required and number of LC required to implement one LFSR. Now we explain CC.

LFSR operators considered in this design scenario are duplicated and as consequence we do not take into account their scheduling. These operators are precisely organized in specific structures described in [46]. Any kind of LFSR implemented is used and re-used by distinct functions through the whole chain. For a particular parametrisation dedicated

to a definite standard, one LFSR is specific to an operation and it is only used in that configuration. So, in that context, we do not focus on a basic execution time of one LFSR as long as LFSR is able to execute different standards under the constraint of each one. In the case of Scrambler, CRC or CC, we replace LFSR structures by their exact replicas based on LFSR COs of same size; the implementation is immediate. As a consequence, LFSR are not defined with a specific execution time. Each and every LFSR designed differs from more or less additional logic elements. For the same *size*, they implement the same number of registers as a consequence, we can estimate that they have basically the same execution time equal to 1 clock cycle. As a result we put CC of each LFSR equal to 1. Since a single operator or a combination of different operators is called only once to perform the required functionality, hence we put NoC equal to 1. Putting 1 for CC and NoC makes the cost function fairly simple.
After putting the costs , we run the optimization algorithm on the graph of the Fig. 7.3. We get the final cost equation as follows:

$$\begin{aligned} \mathbf{C_{LTE1,LTE2,WiFi1}} &= Cost_{14ER+8R+2R+4RG} \\ &= 14 \times 16 + 8 \times 10 + 2 \times 10 + 4 \times 10 \\ &= 364 \end{aligned} \tag{7.1}$$

Optimization process selects *14ER, 8R, 2R and 4RG* nodes that give minimum cost to implement this tri-standards system.
Let us make this example a bit complicated. We introduce the decomposition of higher operators e.g. 22ER into smaller/basic operators. Decomposition of 22ER is shown in the Fig. 7.4. The higher operators i.e. ER-LFSR can be decomposed into R-LFSR and Logic Cells (R-LFSR Mapping) and similarly R-LFSR can be decomposed into RG-LFSR and RF-LFSR as discussed earlier. In Fig. 7.4, decomposition of 22ER to the basic operators is shown and now we add this decomposition into Fig. 7.3. This results in quite a complicated scenario. After addition of this decomposition we run again the optimization algorithm to find the operators with the minimum cost. In this case equation (7.1) becomes:

$$\begin{aligned} \mathbf{C_{LTE1,LTE2,WiFi1}} &= Cost_{14ER+2R+4RG} \\ &= 14 \times 16 + 2 \times 10 + 4 \times 10 \\ &= 284 \end{aligned}$$

Optimization process selects *14ER, 2R and 4RG* nodes that give minimum cost to implement this tri-standards system. It is to be noted that the values used in above equations are from independent LFSR operators i.e. from Table 7.1.
Let us explain the results. In the second part of the example the operators selected are not exactly the same as in the first part. The first operator i.e. 14ER is same in both cases. The operator 8R is not selected. This is because of the fact that when we include the decomposition, 8R can be implemented by the 2R and this operator i.e. 2R is selected. Finally operator 4RG is selected which is the same as in the first part of the example.

It should be noted that here we have considered three modes (2×LTE +1×WiFi) only to elaborate our chosen example. The impact of the CO technique is really noticeable

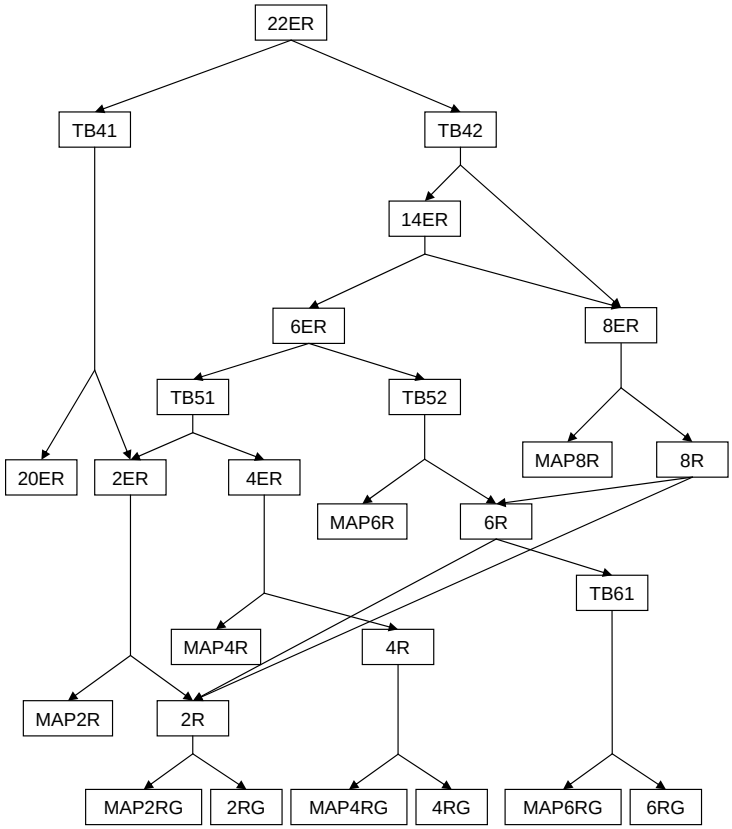

Figure 7.4: Decomposition of 22ER into smaller/basic operators

with more than 3 modes. However, in our specific case the equivalent Velcro method required 336 logic cells (reference percentage of 100%). The *rough* implementation of the ER-LFSR8 (22ER) costs 352 (22×16) logic cells (105%). The implementation of the combination of 14 ER, 2R and 4 RG costs $14 \times 16 + 2 \times 10 + 4 \times 10 = 284$ logic cells (84%).

The first results of the 22ER are over the cost of the Velcro Method and hence justify the enforcement of the optimization process in our case. By using the breakdown of operators in second half of the example we get a reduction of 21% in the cost for designing the same system. These results prove the efficiency of the optimization process in the framework of the CO technique with a gain. Moreover, the results of running the optimization algorithm validates our implementation results.

7.5 Conclusions

In this chapter we provided a detailed treatment of the 2^{nd} potential candidate of COs i.e. LFSR operator that can be used to perform various tasks in an SDR equipment. We described our developed LFSRs which is one of the first implementation of in the context of techniques of parametrisation under the CO approach. We explained the redesign the classical LFSR operators in such a way to be able to provide various functionalities for

multi-standard SDR systems. To evaluate the complexity and speed performances of these operators, we have considered their implementation on ALTERA's Cyclone II FPGA devices. Compared to a Velcro operators, our developed operators presented an important gain in terms of number of logic cells' saving. Graphs of this comparison can be found in [18, 46].

We applied our graphical approach for the design of multi-standard systems. We illustrated our design approach through a simplified sub-design and by running the optimization algorithms on subpart of the graph we proved that different common operators chosen depending upon the considered scenarios give efficient solutions as compared to *Velcro* methods. These results have been accepted for publication in [47].

In the next chapter we present another CO for the channelization of multi-standards SDR systems.

Chapter 8

Case Study: FRMFB as Common Operator

Contents

8.1 Introduction

One of the key issues in designing SDR's multi-standards equipments is the development of efficient channelizers that extract individual channels from the digitized wide-band input signal. In this chapter we describe various filter bank techniques for channelization and compare them to find the best filter bank for channelization. We build on the already presented concept of theoretical approach based on graphical formalism to design multi-standard SDR systems. Based on the general graph models for SDR systems, we focus on the channelizer's branch and draw sub-graph for channelization's task. We explain how we can integrate different channelizers in theoretical approach based on graphical formalism. We illustrate the potential of theoretical approach for optimizing multi-standard SDR

systems by considering realistic examples of channelizers for SDR systems that extract 5 GSM and 5 W-CDMA channels from a wide-band input. Results show that frequency response masking (FRM) technique is most suitable candidate as compared to others, for future SDR channelizers. We discuss the possibilities of considering FRM filter bank (FB) as a common operator. A conclusions's section ends this chapter.

8.2 Filter bank techniques for SDR channelizers

In the context of SDR architectures, one of the challenging issue is the extraction of individual channels from the wide-band digitized signal. To extract multiple channels an obvious approach is to have a separate channelizer for each channel. This so-called per-channel (PC) approach does not look very elegant, although PC approach is an efficient approach, if the number of channels to be received is less [49]. The complexity of the PC approach is directly proportional to the number of channels. Hence the PC approach is not efficient when the number of channels is large. In this section we explore various FB techniques for channelization along with their merits and demerits.

8.2.1 Discrete Fourier transform filter bank

Discrete Fourier Transform Filter Bank technique (DFTFB) is the most popular of all. Efficient implementations of a channelizer using DFTFBs are available in literature [49, 50]. A uniform DFT filter bank can be realized by implementing one low-pass filter and a corresponding modulator such as DFT. Thus instead of implementing N separate channel filters, a single low-pass filter followed by DFT is only required [49]. The limitations of DFTFBs for SDR receiver applications are summarized below [49]:

1. DFTFBs cannot extract channels with different bandwidths. This is because DFTFBs are modulated filter banks with equal bandwidth of all bandpass filters. Therefore, for multi-mode receivers, distinct DFTFBs are required for each standard. Hence the complexity of a DFTFB channelizer increases linearly with the number of received standards.

2. Due to fixed channel stacking, the channels must be properly located for selecting them with the DFT filter bank. The channel stacking of a particular standard depends on the sample rate and the DFT size. To use the same DFT filter bank for another standard, the sample rate at the input of the DFT filter bank must be adapted accordingly. This requires additional sample rate converters (SRCs), which would increase the complexity and cost of DFTFBs.

3. If the channel bandwidth is very small compared to wide-band input signal (extremely narrow-band channels), the prototype filter must be highly selective resulting in very high-order filter. As the order of the filter increases, the complexity increases linearly. Also the DFT size needs to be increased.

8.2.2 Goertzel filter bank

A filter bank channelizer based on modified Goertzel algorithm was proposed in [49] as a solution of second and third problems of the DFTFB. We call this architecture as Goertzel

filter bank (GFB). But since the GFB is also a type of modulated filter bank, the first problem remains unsolved.

In addition to the above three limitations reported in [49], there is a fourth limitation with the DFTFB and the GFB, related to the reconfigurability of the SDR receiver, which can be stated as follows: *The reconfigurability of an SDR receiver must be accomplished by reconfiguring the same filter bank for a new communication standard instead of employing separate filter banks for each standard.* The GFB in [49] is incapable of achieving this reconfigurability.

8.2.3 Hybrid filter bank

A FB based on a combination of polyphase filter bank and DFT modules has been proposed in [51]. The computational complexity of this hybrid FB is less when compared to conventional DFTFBs. However the four limitations mentioned above are not overcome in [51].

8.2.4 Multi-mode DFT filter bank

A multi-mode channelizer that has two stages of DFTFBs and efficient sample rate converter has been proposed in [191]. In this architecture, the front end DFTFB has fixed number of channels, but the passband supports overlap with each other considerably resulting in easier isolation of channels with center frequencies of successive bandpass filters. The outputs of the front end DFTFBs are then fractionally decimated using SRCs. These decimated outputs are fed to the back end DFTFBs. Since the sample rate is considerably lowered in the back end, the DFTFBs at the back end need to operate only at low speed for either fixed or variable number of channels. The drawback of this architecture is that, hardware optimization can be done only for the front end because the back end needs to be changed according to the new communication standard. Another limitation of the channelizer in [191] is its inability to extract channels of different bandwidths (first limitation discussed earlier).

8.2.5 Modulated perfect reconstruction filter bank

In [192], a channelizer based on modulated perfect reconstruction bank (MPRB) has been proposed. The computational complexity of the MPRB is significantly reduced compared to DFTFBs. Also the MPRB can be used for the channelization of signals of unequal bandwidths. A new method for the efficient designing of the MPRB is also proposed in [192]. But when the number of channels to be extracted increases, the computational complexity of the MPRB is larger than the polyphase implementation of DFTFBs. In addition, the channelizer in [192] has the same disadvantage of the methods in [51, 191]: distinct FBs are required for each communication standard.

8.2.6 Tunable pipelined frequency transform filter bank

A pipelined frequency transform (PFT)-based technique has been modified in [193] to develop a channelizer known as tunable PFT (TPFT). This technique allows splitting of frequency bands of a given spectrum of frequencies into a low frequency and high

frequency bands successively at different stages. A PFT divides the wide-band signal into two frequency bands using efficient interleaved connection of complex up and down converters. In TPFT, two levels of tuning are done, a coarse tuning at the PFT level and a fine tuning using another complex up/down converter assisted by a numerical controlled oscillator (NCO) [193]. This channelizer [193] suffers from the drawback that it can only extract signals whose channel spacing are related by a factor-of-two

8.2.7 Tree structured quadrature mirror filter bank

A reconfigurable channelizer using tree structured quadrature mirror filter bank (TQMF) has been proposed in [194]. This channelizer consists of different stages of TQMF, splitting the frequency band into a high and low frequencies at quadrature frequency in each stage. The desired frequency bandwidth is obtained at an appropriate stage. The channelizer proposed in [194] offered reconfigurability at architectural and filter levels of operations. The main drawback of the method in [194] is the delay in obtaining the desired output due to multistage filtering and decimation processes. Also, complexity of TQMF approach in [194] can be further reduced by doing decimation before filtering employing polyphase decomposition.

The channelizer in [194] also suffers from the drawback that it can only extract signals whose channel spacing are related by a factor-of-two. This constraint is imposed by the power-of-two sub-band stacking adopted in these architectures. Another problem of the methods in [193] and [194] is that, as the sub-band decomposition tree extends, the word-length of the output increases linearly and finite precision multiplication would introduce truncation error, which propagates along the tree. Also methods in [193, 194] do not satisfy the reconfigurability principle of SDR as these methods fail to address fourth limitation discussed earlier. It can be noted that the methods in [51, 191, 192, 193, 194] do not offer an efficient trade off between complexity and reconfigurability.

8.2.8 Frequency response masking filter bank

In [195], a reconfigurable filter based on the frequency response masking (FRM) technique [52] was proposed. In [48] a new reconfigurable filter bank based on the FRM approach is proposed called FRM filter bank (FRMFB). Authors have modified the original FRM approach to achieve following advantages: (1) incorporate reconfigurability at the filter level and architectural level, (2) improve the speed of filtering operation and (3) reduce the complexity. The FRM approach has been modified in [48] for improving the speed of operation and to reduce the complexity of the FRM based filter architectures. The proposed FRM based channelizer is able to overcome the limitations 1 to 3 mentioned earlier in this section. The proposed FRM approach offers reconfigurability both at the architectural level and also at the filter level while ensuring low complexity. FRMFB become the ultimate choice for future multi-standard SDR equipment because of the important feature of reconfigurability. Further details can be found in [48].

8.2.8.1 FRMFB technique

Many DSP applications can benefit from the presence of highly-selective linear phase FIR filters. Unfortunately, such filters are historically very complex. Fortunately there is a reduced-order FIR design strategy that provides a means of synthesizing a steep-skirt filter. The method, called *frequency masking*, was introduced in 1986 by [52]. The architecture of a frequency-masked FIR is shown in Fig. 8.1 and consists of the following subsystems:

- Prototype filter: $H_1(z^M)$
- Complement of $H_1(z)$: $z^{M(N_a-1)/2}$
- Upper masking filter: $H_{FMA}(z)$
- Lower masking filter: $H_{FMC}(z)$

Further details about FRM can be found in [52, 196].

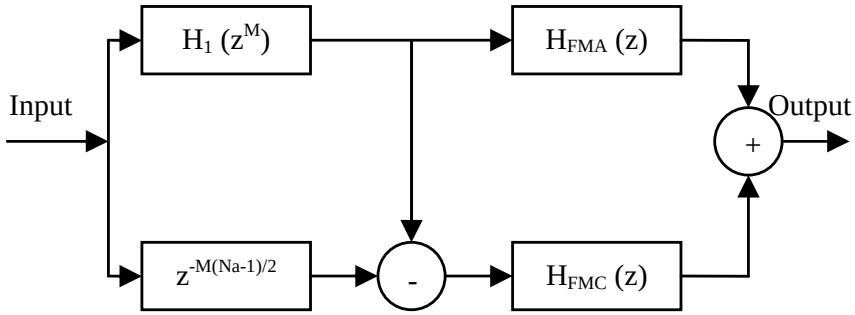

Figure 8.1: FIR filter architecture based on FRM technique

8.2.8.2 Graph model for multi-standards SDR channelizers

The general graph explained in 3 is originally planned to support the complete design of a multi-standard SDR system including all the signal processing elements it contains as in [118]. Here we concentrate only on channelizers' branch. To implement channelizers for a single/multi-standard system we have different alternatives available as shown in Fig. 8.2. The box in the gray color indicate that other communication function are there but ignored at present. In order to implement the optimization procedure of [20, 197], we need to have the costs for each box of Fig. 8.2. These costs can be e.g., installation cost, execution cost, area, number of gates etc. as explained in detail in 4. In design scenarios, presented in the next section we consider that the cost is proportional to the number of multipliers and adders.

8.3 Complexity analysis of filter banks for SDR channelizers

In this section we present two design scenarios based on different FRMFB architectures, that can aid in explaining our approach to design multi standards systems. We perform the complexity analysis of PC approach, DFTFB and FRMFB for channelization.

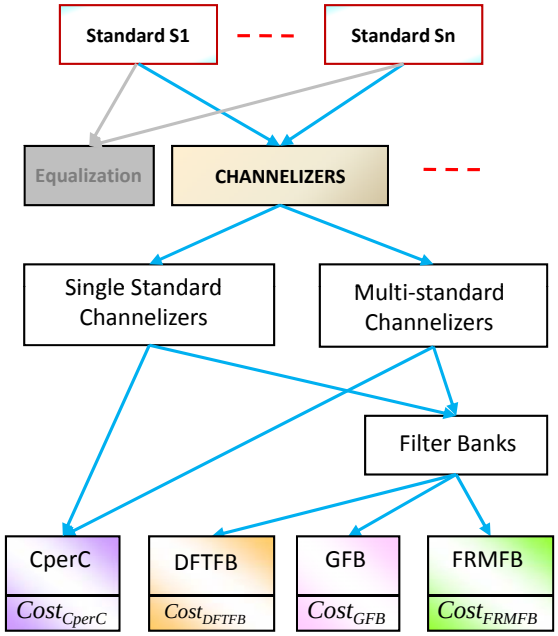

Figure 8.2: Channelizers for single/multi-standard SDR

8.3.1 Design scenario #1

Let us consider the case of dual channel extraction. We replace the standards S1 and S2 in Fig. 8.2 by GSM and WCDMA. The objective is to extract 5 channels of GSM (200 kHz) and 5 channels of WCDMA (3.84 MHz) from a 20MHz wide-band input (sampled at 40 MHz) under abundantly occupied spectrum scenario. The distributions of the GSM channels and WCDMA channels are shown in Fig. 8.3 and Fig. 8.4 respectively. The architecture for FRMFB for extracting the above 5 GSM and 5 WCDMA channels is shown in Fig 8.5.

Modal filter specifications:

- Passband edge (f_p): 2 MHz
- Stopband edge (f_s): 4 MHz
- Passband ripple (δ_p): 0.1 dB
- Stopband attenuation: (δ_s)-60 dB

The optimum filter length, N, of the filter can be calculated by the formula ((8.1)) presented in [198].

$$N = -\frac{2log_{10}(10\delta_p\delta_s)}{3(f_s - f_p)} - 1 \tag{8.1}$$

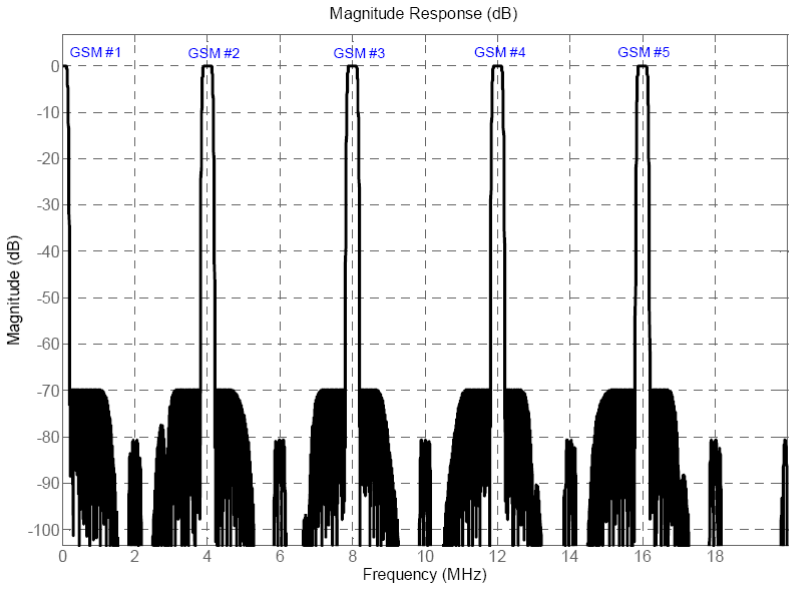

Figure 8.3: GSM channel distribution

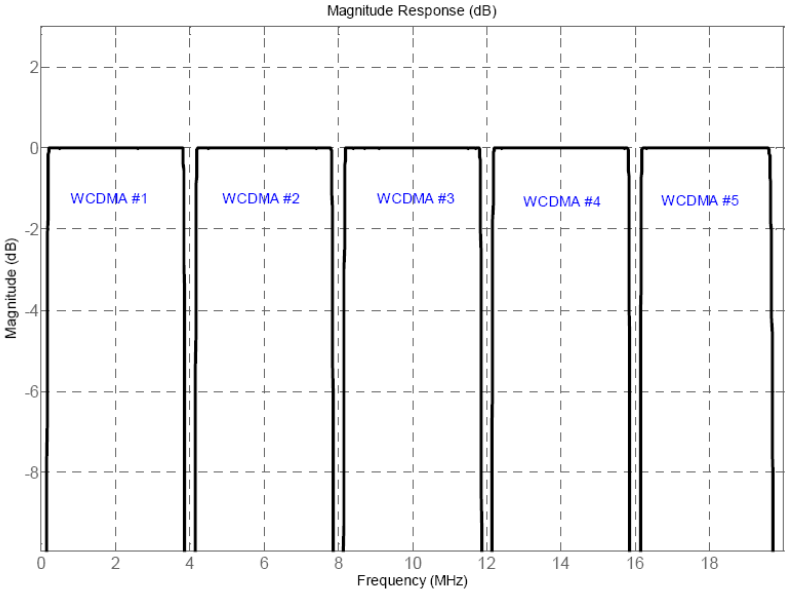

Figure 8.4: WCDMA channel distribution

Using the specifications of the modal filter the optimum length for filters is given in Table 8.1. The complexity associated with each filter for the FRMFB in Fig. 8.5 is also mentioned in Table 8.1.

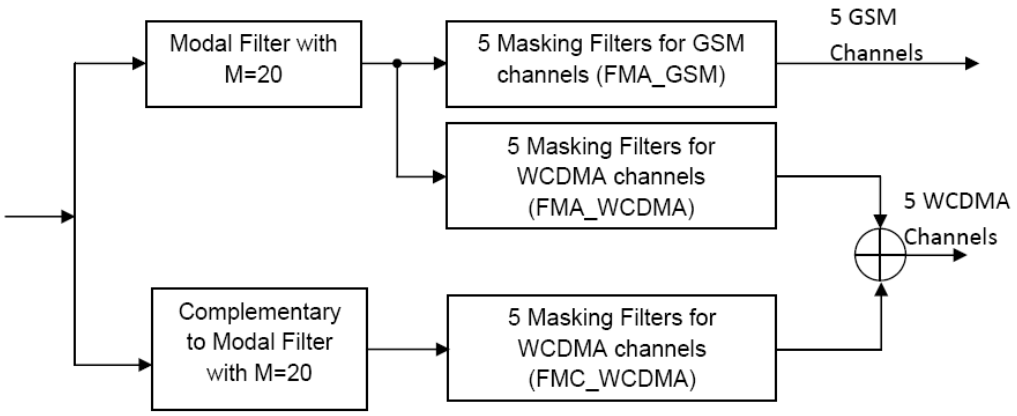

Figure 8.5: Architecture of FRMFB - Design-1

Table 8.1: Computational complexity of FRMFB for the extraction of 5 GSM and 5 WCDMA Channels - Design-1

Name of Filter	**No. of Taps**	**No. of Adders**	**No. of Multipliers**
Modal Filter	81	41	80
FMA_GSM	600	$300 \times 5 = 1500$	$599 \times 5 = 2995$
FMA_WCDMA	250	$125 \times 5 = 625$	$249 \times 5 = 1245$
FMC_WCDMA	250	$125 \times 5 = 625$	$249 \times 5 = 1245$
Total complexity	-	2791	5565
Complexity of PC/ DFTFB	1400	7000	13990

8.3.2 Design scenario #2

In this design scenario objective is the same as in design scenario 1 i.e. to extract 5 channels of GSM (200 kHz) and 5 channels of WCDMA (3.84 MHz) from a 20MHz wide-band input. However we change the architecture of FRMFB. This new architecture of FRMFB for extracting the above 5 GSM and 5 WCDMA channels is shown in Fig 8.6.

New Modal filter specifications:

- Passband edge (f_p): 1 MHz
- Stopband edge (f_s): 2 MHz
- Passband ripple (δ_p): 0.1 dB
- Stopband attenuation (δ_s): -60 dB

Using the specifications of the modal filter the optimum length for filters is mentioned in Table 8.2. The complexity associated with each filter for the FRMFB in Fig. 8.2 is also given in Table 8.2. The complexity of PC/DFTFB is given in the last row of Table 8.1 and Table 8.2 for both scenarios. For PC and DFTFB, the complexity figures are

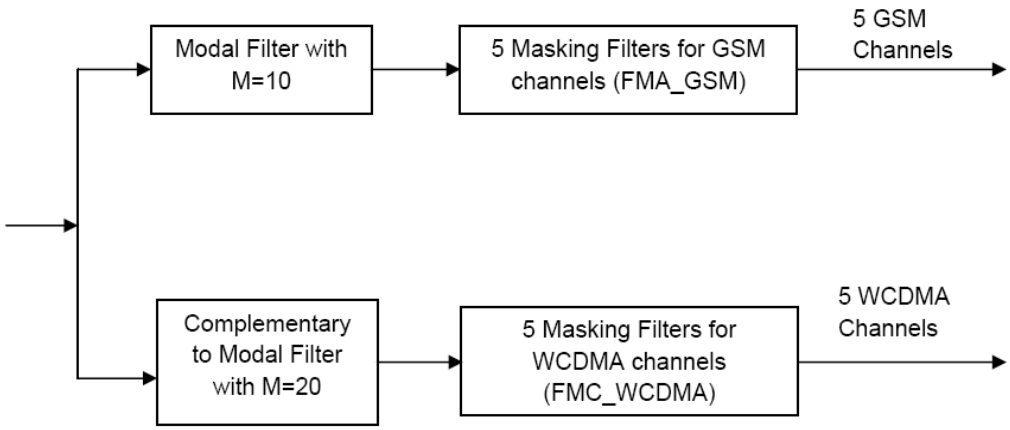

Figure 8.6: Architecture of FRMFB - Design-2

Table 8.2: Computational complexity of FRMFB for the extraction of 5 GSM and 5 WCDMA Channels - Design-2

Name of Filter	**No. of Taps**	**No. of Adders**	**No. of Multipliers**
Modal Filter	181	91	180
FMA_GSM	300	$150 \times 5 = 750$	$299 \times 5 = 1495$
FMC_WCDMA	250	$125 \times 5 = 625$	$249 \times 5 = 1245$
Total complexity	-	1466	2920
Complexity of PC/ DFTFB	1400	7000	13990

considered to be same because there is no much difference in their complexities in the presented scenarios compared to FRMFB complexity.

8.4 Optimization of channelizers

Let us describe the optimization process for SDR channelizers. We needed costs to put on the nodes of graph model. These costs are found in the 8.3 in terms of number of multiplier and adders. These costs are to be used in the optimization process. In the design scenarios of section 8.3, we have used only the PC, DFTFB and FRMFB. The optimization process consists in finding the solution to the following equation:

$$\mathbf{C_{Channelizer}} = \min\left(Cost_{CPerC}, Cost_{DFTFB}, Cost_{GFB}, Cost_{FRMFB}\right) \quad (8.2)$$

Let us explain the cost computation of FRMFB node. The cost of the FRMFB node can be computed by equation (4.10) duplicated below:

$$Cost_{level} = \left(\sum_{i=1}^{i=m} CC_i NoC_i\right) \times NoC_{i+1} + \sum_{i=1}^{i=m} BC_i.N_i \quad (8.3)$$

$N_i = 1$ since cost involves only those nodes that are present in the system. Considering FRMFB node only we put $NoC_{i+1} = 1$, so equation (8.3) becomes:

$$Cost_{FRMFB} = \Big(\sum_{i=1}^{i=4} CC_i NoC_i\Big) \times 1 + \sum_{i=1}^{i=4} BC_i \tag{8.4}$$

For FRMFB node, equation (8.4) becomes:

$$\begin{aligned} Cost_{FRMFB} &= Cost_{ModalFilter} + Cost_{FMA_{GSM}} + Cost_{FMA_{WCDMA}} \\ &+ Cost_{FMC_{WCDMA}} \end{aligned} \tag{8.5}$$

The costs for different components for FRMFB in terms of number of adders and multiplier are expressed in Table 8.1.

$$\begin{aligned} Cost_{FRMFB} &= \big((41 + 1500 + 625 + 625) \times CC_{Adder} \\ &+ (80 + 2995 + 1245 + 1245+) \times CC_{Multiplier}\big) \\ &+ BC_{Adder} + BC_{Multiplier} \end{aligned} \tag{8.6}$$

Considering the implementation of FRMFB on Altera Stratix-II FPGA family, same BC and CC costs for the adder and multiplier can be used as given in Table 6.3.

$$Cost_{FRMFB} = \big(2791 \times 2.37 + 5565 \times 5.18\big) \times 1 + 26 + 104 = 35571.37 \tag{8.7}$$

Similarly, the cost of PC/DFTFB channelizer can be computed as follows:

$$\begin{aligned} Cost_{PC/DFTFB} &= \big(NoC_{Adder} \times CC_{Adder} + NoC_{Multiplier} \times CC_{Multiplier}\big) \\ &+ BC_{Adder} + BC_{Multiplier} \end{aligned} \tag{8.8}$$

or,

$$\begin{aligned} Cost_{PC/DFTFB} &= \big((7000) \times CC_{Adder} + (13990) \times CC_{Multiplier}\big) \\ &+ BC_{Adder} + BC_{Multiplier} \end{aligned} \tag{8.9}$$

or,

$$Cost_{PC/DFTFB} = \big(7000 \times 2.37 + 13990 \times 5.18\big) + 26 + 104 = 89188.20 \tag{8.10}$$

The design scenario presented in section 8.3.2, where we just change the FRMFB architecture, gives us the advantage in terms of further lower complexity. The cost of FRMFB in this case will be as follows:

$$Cost_{FRMFB_{\#2}} = \big(1466 \times 2.37 + 2920 \times 5.18\big) \times 1 + 26 + 104 = 18730.02 \tag{8.11}$$

Result of running the optimization chose FRMFB to be a good candidate for SDR channelizers. These results may seem to be obvious since FRMFB has much lower complexity as compared to other approaches as shown in Table 8.1 and Table 8.2. However, if we include one more standard e.g. WiFi, result will not be obvious at all. This is because of the fact that WiFi requires the use of FFT for the OFDM (de)modulation and in this

case our optimization algorithm may choose DFTFB as best candidate. The reason for this is that now two communication blocks i.e. Channelization and OFDM can use the FFT. Hence in this case our optimization tool will find the global optimality. However in the presented scenario FRMFB is clearly the best choice.

This is almost what is always done. Researchers are proposing local optimums on their sub-parts of interest. We aim at showing that their local optimum may be in contradiction to the global optimum. As our tool results aim at finding global optimum so in the next chapter a case study is presented to elaborate this.

8.5 Conclusions

In the context of multi-standard SDR design, we elaborated the theoretical approach to find the common operators based on graphical formalism. We illustrated our design approach through a simplified sub-design concentrating only on the channelization part of SDR. To evaluate the complexity of various filter bank techniques, we performed a quantitative analysis based on the underlying mathematical equations/formulas in the two design scenarios. Compared to a Velcro approach (PC approach) and the DFTFB approach, the FRMFB approach resented an important gain in terms of number of multiplier and adders. We can foresee from the complexity analysis performed in design scenarios, that FRMFB can be chosen for a multi-standard, multi-channel channelizer as it offers minimal cost. Consequently, this reinforces the interest of FRM technique for SDR design.

The full potential of our theoretical approach cannot be fully exploited as the sub-designs chosen are fairly simple ones (for illustration purposes). However if we include one more standard to be supported, say WiFi, the design scenario will become quite interesting. This is because of the fact that now DFTFB will be of greater importance as the FFT used in DFTFB can also be used by the OFDM (de)modulation, which is not the case in the presented scenario. This scenario is presented in next chapter in which we design a more complete hypergraph having more COs supporting multiple air interfaces.

Chapter 9

Case study of a complex graph

Contents

9.1 Introduction

In this chapter we try to summarize/present the combined results of COs discussed in part-III. We try to integrate the two common operators i.e. DMFFT operator presented in chapter 6 and FRMFB operator presented in Chapter 8 in a more complete scenario that can be considered as a step towards achieving the ultimate aim of getting a complete and optimized graph model of multi-standards system involving the common operators.

One of the aim of this thesis was to use the COs in the graph and prove the worth of using these operators using the theoretical approach under the umbrella of common operator's technique for parametrisation. This was done in last three chapters. But our approach aims at thinking the whole design at a complete system level, so in this section we try to look from the system level point of view. We explain how we can use these operators described in previous chapters to design a multi-standards SDR equipment. So far if we look at the development of these COs, these COs were developed by looking at the functions that can be grouped together. Here we try to seek for the integration of these COs into a single graph. This graph of multi-standards SDR system, that we try to build and then optimize by using the graph optimization techniques as discussed in chapter 5 consists of designing a tri-standard system. This system consists of WiFi, WiMAX and UMTS. WiFi and WiMAX are OFDM based systems and UMTS is W-CDMA based one. These systems consists of a wide range of different types of functionalities that are to be carried out to process the data packets. We want to build a more detailed graph incorporating more than one common operator discussed so far as this may permit to find a global minimum

in the design that can be different from the local minimum found for each sub-part of the design. Usually the optimization research is focused on sub-parts and consequently finds local optimum. This is real worth of this PhD to propose a way to find the global optimum.

9.2 Common operators

Let us start our discussion with Fig 9.1. This figure consists of two blocks. Block 1 corresponds to DMFFT operator and Block 2 corresponds to FRMFB operator.

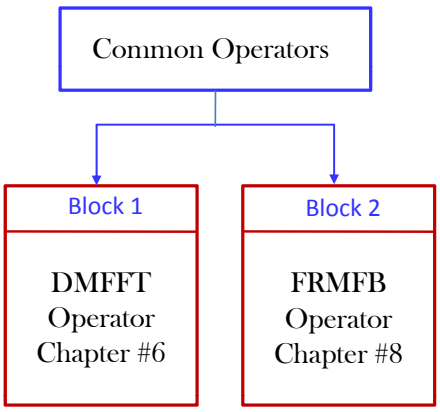

Figure 9.1: An generalized view of common operators

Till now the work presented corresponds to the individual blocks as detailed in Chapter 6 &8. But here we try to look for the possibilities that whether we can use the CO of one block to perform the functionality of other block or not. For example, the DMFFT operator can be used for OFDM (de)modulation and some steps of RS decoding but the same DMFFT operator can be used in the *filter bank branch* of channelizers i.e. for the DFTFB. Although in Chapter 8 we discussed that FRMFB is the best option for the channelization, but the above scenario presents a very interesting case as in above scenario DFTFB cannot be neglected since it may give us a global optimal as it uses the already present DMFFT operator for OFDM (de)modulation and RS decoding. Our objective is to find the global optimum, so above scenario describes exactly our philosophy. Here we look from a top level, the possibilities of using the same DMFFT for the channelization as well.

Let us explain Fig. 9.1 in more detail. DMFFT can do the OFDM demodulation and some steps of RS decoding which is a big achievement in itself. This was comprehensively detailed in chapter 6. The FRMFB is best for the channelization as detailed in chapter 8. Now what makes the DMFFT even more interesting is the arrow that is coming from the channelizer's branch i.e. from DFTFB. In this case we may reach at the result; for block 1 DMFFT was choice, for block 2 FRMFB was the choice but because of arrow coming from block 1 to Block 2 DMFFT may result in global optimal solution which is to be checked by the optimization tool.

9.3 Graph model

Let us develop a graph of the system. Consider a more complete graph of a multi-standards SDR system to be optimized. We consider the possibility that the tri-standards system presented in Fig. 9.2 requires only the functionality of OFDM, RS decoding and channelizers. In this way we will be able to combine the results of two COs i.e. DMFFT and FRMFB. The best would be to imply all the functions of the whole protocol stack of all three standards (like scrambling, interleaving etc.) but it is far beyond the possibility of this PhD work and the scope of this thesis. In addition, representing a too complicated graph would not be of much help (for understanding purposes) regarding the explanations that we wanted to give in this thesis.

Let us draw the graph model for this system. We draw this model by using our developed GUI as described in section 4.5. This GUI tool can help us in drawing graph models and associating the respective cost parameters as mentioned earlier. The result is shown in Fig. 9.2. As evident from the Fig 9.2 the visibility is not very clear, so in Fig. 9.3 we present an enlarged version of the same figure. As in previous chapters this graph also does not contain all the operations of full protocol layers of the three standards (which would have been impossible to obtain during the thesis and would not facilitate the explanations necessary here), but it is reduced to OFDM demodulation, RS decoding and channelization functions.

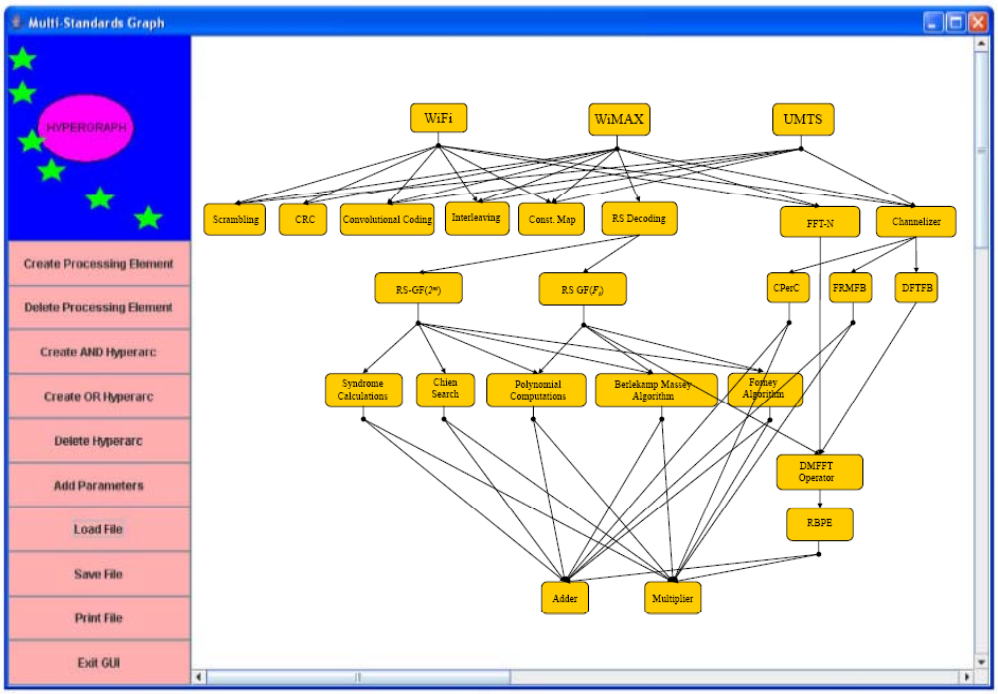

Figure 9.2: Screen shot of graph drawn in GUI

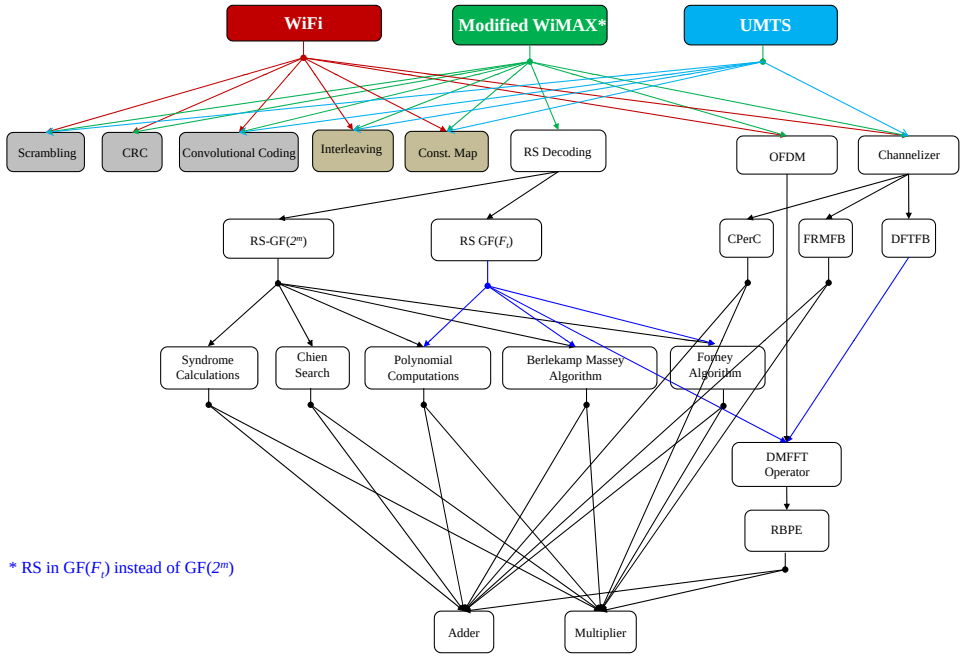

Figure 9.3: Zoomed version of graph in Fig. 9.2

9.4 Cost issues

Let us put the costs. These costs can be obtained from the complexity evaluation as given for WiFi and UMTS in the Appendix B. The costs used in the previous chapters are not compatible. By non compatible we mean that they are not in the same unit i.e. the costs for the DMFFT are in terms of ALUTs, costs of LFSRs are in terms of Logic Cells and costs of channelizers are in terms of multiplications and additions. So we try to express the costs in a common unit i.e. for each block of the systems either we will express the cost in terms of LC or ALUTs or adders and multiplier. This will be explained later when we will put the costs on the blocks. The costs that are to be put on the blocks are based on the actual complexity evaluation of the systems.

Now we describe the cost parameters of different blocks of Fig. 9.3 that are mentioned in Table 9.1. Some of the costs are taken from Table 6.3. The BC/CC costs for FRMFB and DFTFB are taken from [48], where the costs are given in number of gates. These costs are then converted into number of ALUTs. This conversion can be achieved by the rule $1\ ALUT \approx 1.5 \times LC$ and $1\ LC = 30\ to\ 40\ Gates$ on a Virtex FPGA family. The costs associated with arrows i.e. NoC are taken from Table 6.4 and Table 8.1. The DFTFB can use the complexity of DMFFT operator which is necessary for OFDM and some steps of RS decoding, so we expect that in the overall design channelization may be performed

Table 9.1: Cost parameters for different blocks of Fig. 9.3

Name of Function	BC (ALUTs)	CC (ns)
DMFFT	4857	5.5
RBPE	514	5.18
Re-Adder	26	2.37
Re-Multiplier	104	5.18
DFTFB	11868	77
FRMFB	7938	38.67

by DFTFB. In the next section we see whether our tool permits to validate it or not.

9.5 Optimization results

After putting the costs of PEs and arrows, we run our optimization tool. Results of running the optimization tool selects RPBE, PolyCompt (short for polynomial computation), Berlekamp (short for Berlekamp Massey Algorithm) and Forney (short for Forney Algorithm) to implement the system with minimum cost. The results at different index of iterations are shown in Fig. 9.4.

Let us explain these results. The cost for the system shown in Fig.9.3 can be written as follows:

$$\mathbf{C_{System}} = Cost_{RS_{Decoding}} + Cost_{OFDM} + Cost_{Channelizer} \tag{9.1}$$

Let us find the cost of each function of equation (9.1). The cost of the RS decoding (also mentioned earlier) can be computed by equation (4.10) duplicated below:

$$Cost_{level} = \Big(\sum_{i=1}^{i=m} CC_i NoC_i \Big) \times NoC_{i+1} + \sum_{i=1}^{i=m} BC_i.N_i \tag{9.2}$$

$N_i = 1$ since cost involves only those nodes that are present in the system. As we are considering RS decoding node only, hence $NoC_{i+1} = 1$ and, equation (9.2) becomes:

$$Cost_{RSDecoding} = \Big(\sum_{i=1}^{i=5} CC_i NoC_i \Big) \times NoC_{i+1} + \sum_{i=1}^{i=5} BC_i \tag{9.3}$$

Using the first option to perform RS decoding, equation (9.3) becomes:

$$\begin{aligned} Cost_{RS-GF(2^m)} &= Cost_{Syndrome} + Cost_{Chien} + Cost_{Poly} \\ &+ Cost_{Berlekamp} + Cost_{Forney} \end{aligned} \tag{9.4}$$

or,

$$\begin{aligned} Cost_{RS-GF(2^m)} &= \big((4096 + 2048 + 392 + 512 + 16) \times CC_{Multiplier} \\ &+ (4096 + 2048 + 384 + 512 + 8) \times CC_{Adder}\big) \times 1 \\ &+ BC_{Multiplier} + BC_{Adder} \end{aligned} \tag{9.5}$$

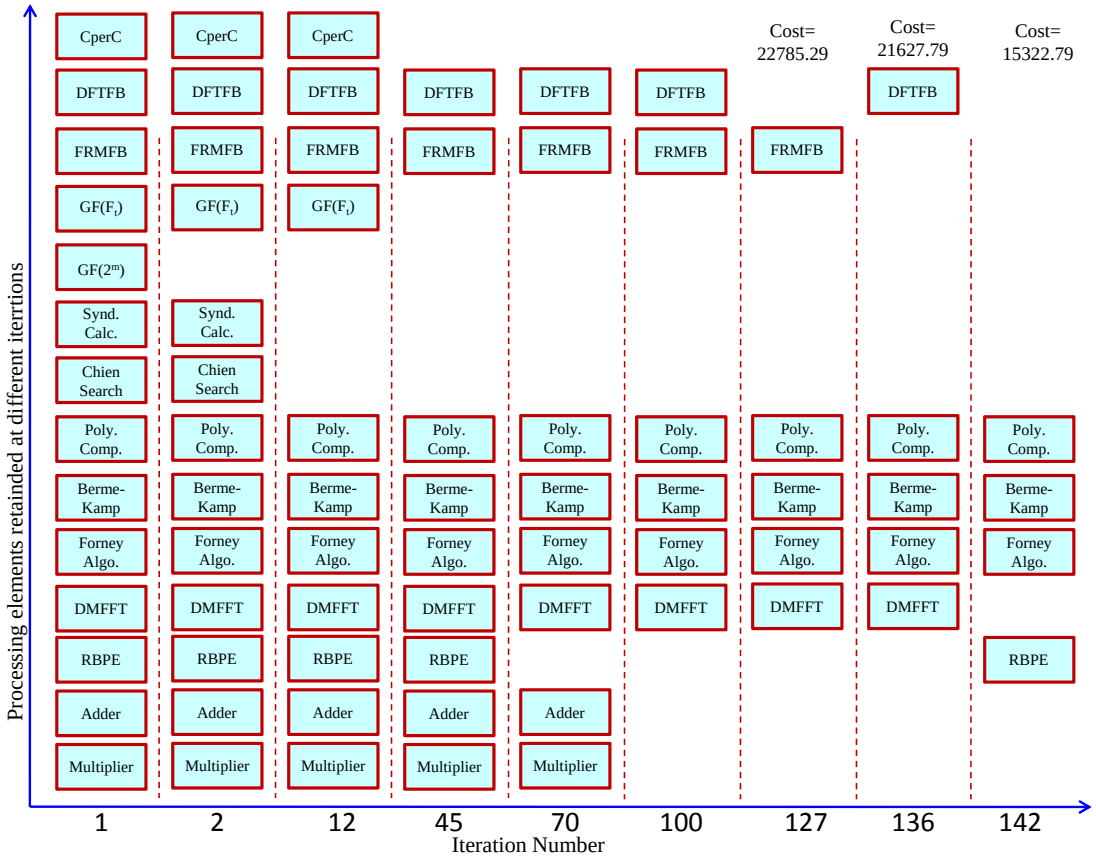

Figure 9.4: Number of operators kept at different number of iterations

or,

$$Cost_{RS-GF(2^m)} = \big(7064 \times 5.18 + 7048 \times 2.37\big) \times 1 + 104 + 26 = 53425.28 \tag{9.6}$$

Using the second option to perform the RS decoding, equation (9.3) becomes:

$$Cost_{RS-GF(F_t)} = Cost_{Poly} + Cost_{Berlekamp} + Cost_{Forney} + Cost_{DMFFT} \tag{9.7}$$

or,

$$\begin{aligned} Cost_{RS-GF(F_t)} &= \big((392 + 512 + 16) \times CC_{Multiplier} \\ &+ (384 + 512 + 8) \times CC_{Adder} + 2 \times CC_{DMFFT}\big) \times 1 \\ &+ BC_{Multiplier} + BC_{Adder} + BC_{DMFFT} \end{aligned} \tag{9.8}$$

or,

$$Cost_{RS-GF(F_t)} = \big(920 \times 5.18 + 904 \times 2.37 + 2 \times 5.5\big) \times 1 + 104 + 26 + 6819 = 13868.08 \tag{9.9}$$

Considering the RS decoding node only, we clearly see that second option i.e. implementation of RS in $GF(F_t)$ gives the minimum cost. So far we have considered the implementation of RS in $GF(F_t)$ by using the DMFFT operator. But DMFFT operator can

itself be implemented by RBPE. Let us find the cost of DMFFT (using equation (4.10)) implemented through RBPE.

$$\begin{aligned} Cost_{DMFFT_{RBPE}} &= CC_{RBPE} \times NoC + BC_{RBPE} \\ &= 5.18 \times 128 + 514 \\ &= 1177.04 \end{aligned} \tag{9.10}$$

This shows that cost of DMFFT when implemented through RBPE is less as compared to implemented alone. Hence using the this implementation, equation (9.7) becomes:

$$Cost_{RS-GF(F_t)} = \big(920{\times}5.18{+}904{\times}2.37{+}2{\times}(128{\times}5.18\big){+}104{+}26{+}514 = 8878.16 \tag{9.11}$$

Let us consider the OFDM demodulation. It requires FFT-N node which is implemented by DMFFT. The DMFFT node will be called to perform this functionality. So cost of the OFDM (de)modulation will be:

$$\begin{aligned} Cost_{OFDM} &= Cost_{DMFFT} = BC_{DMFFT} + CC_{DMFFT} \\ &= 6819 + 5.5 = 6824.5 \end{aligned} \tag{9.12}$$

$$\tag{9.13}$$

But the DMFFT node is already present in case of RS decoding. According to equation (9.10) DMFFT node can be implemented with less cost by using RBPE. Moreover, in the presence of RBPE (in case of RS decoding), BC will not be duplicated and only CC will change, i.e. effectively for OFDM, $BC = 0$ (due to already present RBPE in case of RS decoding) and CC depends upon NoC. To perform ODFM demodulation DMFFT will be called once or RBPE will be called 128 times. Hence cost of OFDM will become:

$$\begin{aligned} Cost_{OFDM} &= Cost_{DMFFT} = BC_{DMFFT} + CC_{DMFFT} \\ &= BC_{DMFFT_{RBPE}} + CC_{DMFFT_{RBPE}} \\ &= 0 + 1 \times (128 \times 5.18) \end{aligned} \tag{9.14}$$

$$= 663.04 \tag{9.15}$$

Reduction in the cost is clearly evident from equation (9.12) to equation (9.14). It is to be noted that in the presence of DMFFT operator in RS decoding the cost of OFDM is reduced because of re-use of already present operator. In addition, if we use the RBPE to implement the DMFFT the cost is even further reduced which will result in an overall reduction in the cost of the design.

Now we explain cost for channelization. It can be performed by any of the three options shown in Fig. 9.3. For the option of FRMFB and DFTFB the cost of channelizer is:

$$\begin{aligned} Cost_{Channelizer_{DFTFB}} &= BC_{DFTFB} + CC_{DFTFB} \\ &= 11868 + 77 = 11945 \end{aligned} \tag{9.16}$$

and,

$$\begin{aligned} Cost_{Channelizer_{FRMFB}} &= BC_{FRMFB} + CC_{FRMFB} \\ &= 7938 + 38.67 = 7976.67 \end{aligned} \tag{9.17}$$

Looking at the costs for channelizers separately it is clear that FRMFB has low cost than DFTFB.

Now we consider the design as a whole. In this case DFTFB can use the DMFFT operator to implement its functionality. This DMFFT is must in case of OFDM demodulation and RS decoding in $GF(F_t)$. In the presence of DMFFT the BC of the DFTFB now reduces to $11868 - Cost_{DMFFT} = 5043.5$ which results in an overall BC of DFTFB (5043.5) which is less than cost of FRMFB (which is 7976.67). It is to be noted that cost of DMFFT has already been taken care of (in case of combined design) by RS Decoding. We know that CC depends upon NoC. If DMFFT is called once and we consider implementation of DMFFT through RBPE, then $NoC = 128$. Hence, cost of DFTFB in the presence of DMFFT becomes:

$$\begin{aligned} Cost_{Channelizer_{DFTFB}} &= BC_{DFTFB} + CC_{DFTFB} \\ &= BC_{DFTFB} + 77 + CC_{DMFFT} \\ &= 5043.5 + 77 + 1 \times (128 \times 5.18) \\ &= 5783.54 \end{aligned} \tag{9.18}$$

$$\tag{9.19}$$

Using equations (9.11), (9.14) and (9.18), equation (9.1) become:

$$\begin{aligned} \mathbf{C_{System}} &= Cost_{RS-GF(F_t)} + Cost_{OFDM} + Cost_{Channelizer_{DFTFB}} \\ &+ 8876.16 + 663.09 + 5783.54 \\ &= 15322.79 \end{aligned} \tag{9.20}$$

In an independent implementation which does not consider the COs and hence there is no sharing of operators, the cost of the system will be:

$$\begin{aligned} \mathbf{C_{System}} &= Cost_{RS-GF(F_t)} + Cost_{OFDM} + Cost_{Channelizer} \\ &+ 13686.08 + 6824.5 + 7976.5 \\ &= 28669.25 \end{aligned} \tag{9.21}$$

Above results show the advantages we get in the overall cost of the system in case of COs' usage. These results confirm our expectation that in the overall design, DMFFT can be chosen because of its complexity sharing. Results of running the optimization tool show that starting from all the nodes it chooses RBPE to implement the system as

indicated in Fig. 9.4. RPBE is the core of DMFFT operator. Selection of RPBE implies that the global optimal solution comes through DMFFT operator. Selection of RPBE implies that for channelizers implemented by DFTFB gives global minimum, although when considering independently FRMFB was the option. In general we can conclude that for global optimality, channelizers' implementation with DFTFB is the choice and not the FRMFB because of the presence of DMFFT operator which is mandatory for OFDM demodulation and some steps of RS decoding. This further reinforces our interest in using DMFFT in OFDM based standards. Let us recall again that we are well aware of the fact that the standards in our example are already established with GF(2^m).

9.6 Conclusions

This chapter summarized the philosophy of finding a solution by architectural exploration proposed in this thesis. It explained the means for a designer to find a global optimum in terms of COs for multi-standard SDR equipment.

In chapter 6 we explained and proved the worth of DMFFT operator. It was shown that DMFFT has a potential to be used as CO especially in OFDM based systems and also in the systems requiring RS coding. In chapter 8, we explored different channelizers and found FRMFB to be a good candidate for CO.

As an attempt towards more complete graph we tried to integrate the DMFFT operator and FRMFB operator in the same graph model.

It is worth mentioning that as long as we were working on separate sub-parts, the results with optimum for those sub-parts or we can say that we were involved in finding the local optimum. But merging the two sub-parts together resulted in the search to find the global optimum for the whole system.

We have not shown bigger/complete graph examples in this thesis because of a lack in terms of implementation figures for all the PEs of all standards' protocol stack (of interest). It was out of the scope of this thesis as the major concentration was on the design methodology proposed in this manuscript. It would have required a lot of development time to develop a protocol stack (in VHDL). It was not affordable spending much time at development instead of focusing on the main research of this thesis.

General Conclusions and Perspectives

This thesis presented the research work carried out in SCEE team at Supélec, Rennes campus, on the optimization of multi-standards software radio equipments using a theoretical approach. The approach that is based on graphical formalism approach falls into COs' technique for parametrisation which is a promising technique in SR context.

General conclusions

We began this manuscript by presenting the evolution of software radio (SR) technology from functional and historical perspectives. After emphasizing on the need for the software radio in **Chapter 1**, we came to the conclusion that to cope with ever increasing demands of users, manufactures and service providers, the focus has been shifted toward SR approach and conventional *Velcro* approach of designing radio terminals and basestations will become obsolete very soon. The motivation for introducing optimization aspects of flexible radio system design is the tremendous interest in reducing the cost, complexity and size of the radio terminal/base station which are required to support such a wide range of existing and future wireless technologies and services. SDR projects' survey in **Chapter 1** showed how the software radio concept is developing and identified key concepts and technologies useful for the practical implementation of a software radio.

Starting with some conceptual considerations we can say that the *parametrisation* is a technique that aims at optimizing the cost-performance trade off while building a multi-standards SDR equipment. In the context of techniques of parametrisation we discussed the *common function* and *common operator* techniques in **Chapter 2**. Two approaches to identify and develop CO technique were explained. We illustrated the theoretical approach of finding common operators in greater details. Theoretical approach involves the identification of an optimal level of granularity, from which a component can be considered as a common communication block enabling its reuse by several applications. The selection of the most appropriate level of granularity helps to balance between economy and computing efficiency. In the context of theoretical approach we addressed the issues of 1)definition of graph, 2) definition and development of the cost function and 3) optimization of developed graph models in subsequent chapters.

In **Chapter 3** we discussed the idea of modeling air interface standards as graphs. We addressed the question of defining a suitable graph model of multi-standard SDR equip-

ments. Starting from the fundamentals of graph theory and after exploring the already present graphs we came up with the idea of modeling the graph as hypergraph to meet our needs. This is because of the dependencies needed for our problem which were not available for other graph types. By doing this we adequately addressed the problem of *definition of the graph*. Based on this definition we described the development of graph theoretic models for multi-standards SDR systems. After having the graph we got a clear picture of various alternatives to implement the SDR equipment. In order to find the best of the available alternatives that balances between cost and economy we turned our graph model into an optimization problem to address the 2^{nd} and 3^{rd} problems posed in chapter **Chapter 3**. To do this one of the many possible solutions was to associate cost parameters with nodes and arcs of the graph i.e. turn the graphs into weighted hypergraphs.

Costs and cost parameters were discussed in **Chapter 4**. Finding these cost parameters and then associating them to graph was a challenging problem. We identified three most appropriate types of cost parameters i.e. BC, CC and NoC, but there can be many more. The identified cost parameters are to be associated with graph entities. Now expressing the costs in terms of identified cost parameters, itself was a real hard task as even with a single cost parameter, there exist hundreds of different possibilities depending upon platforms, target technologies, clock speeds etc. Hence these costs depend upon the choice of digital hardware e.g. DSP, FPGA and ASIC. The problematic for the costs in case of FPGA implementation was solved. The BC for the FPGA implementation can be expressed in LC, LUT, ALUTs or number of gates etc. CC can be considered in terms execution time. However, we did not find a good way to express the costs in case of DSP implementation i.e. the problems of costs in case of DSP is yet to be solved. For designs consisting of FPGAs only, costs' problem is solved but for heterogeneous design consisting of DSPs and FPGAs we need further investigation of the problem. After considering these parameters and associating them with the graph entities, our graph is turned into an optimization problem.

Based on the selected costs associated with different graph entities we formulated our cost/objective function to be solved by optimization techniques presented in **Chapter 5**. We formulated our objective function as multi-objective optimization problem (MOP) also known as multi-performance/vector optimization problem. After having formulated the objective function our next step was to find its solution. This optimization problem could be solved by different techniques. Among these techniques we had the choice to select between the optimal and near optimal solutions. Due to certain limitations, techniques providing optimal solution were no longer feasible and we were left with no choice but to switch to techniques that provide near optimal solution.

Many real-world scientific and engineering MOPs are irregular and hence deterministic search techniques are not suitable for them. To solve these irregular problems, stochastic search and optimization approaches such as Simulated Annealing, Evolutionary Computation Algorithms etc. were developed as alternative approaches that were addressed in **Chapter 5**. Having considered many techniques that give sub-optimal solution we came to the conclusion that Simulated Annealing and Genetic Algorithms are best suited to the kind of the problem that we had at hand. We developed a tool based on these techniques

that can be used to solve our problem. A generic design example was developed in order to compare the three techniques i.e. exhaustive search, simulated annealing and genetic algorithm. Results of the comparison showed that among the three, SA was the best choice to solve our optimization problem. Exhaustive search (with fewer nodes) could only be used for the verification of the results obtained from other techniques.

As examples, we presented three COs namely LFSR, DMFFT and FRMFB that were developed to be used in SDR equipment and applied the selected optimization techniques on sub-parts of real world SDR systems. One aim of this thesis was to investigate/propose new common operators. We presented three potential candidates for COs to be used in multi-standards SDR systems and showed how these CO can play their role in optimization of radio systems. Of course design and development of these operators is a tedious tasks and every operator requires a work of PhD. These operators were developed by different PhD students working in different Labs. We collaborated with these student to find ways to use these operators in our graph model. In this effort some of the operator were specially modified to use them in the graph models e.g. LFSR operators. We addressed the issue of resolving complexity of these operators to be used in our models. In Chapters 6, 7, & 8, we described these three operator in detail and explained their respective architectures and various possible implementations strategies. We introduced some design scenarios and drew graphs for these scenarios. We integrated these operators in graphical models of aforementioned scenarios and ran the optimization algorithms. Results of running these algorithms highlighted the advantages that we may have by using these operators in future SDR systems.

In **Chapter 9** we presented a more complete example involving two COs in order to explain the idea of obtaining a global optimal solution by our approach. Our ultimate aim was to develop complete system graphs and integrate all the available and new COs in them and run the optimizations. Further work is needed to achieve this goal as described in perspectives.

Perspectives

This thesis provided a framework that naturally extends to investigate future research topics of interest among which we mention the following:

- The proposed approach implies the re-use of operators. As a consequence scheduling issues have to be considered. At a first level, as long as an operator may run in real-time whatever the number of operations call it, the problem can be solved by conventional static scheduling facilities or by some RT-scheduling algorithm played by some manager (why not an RTOS on a processor) of the equipment. At a second level, if the operator is over-used, it should be considered in the methodology to duplicate this operator. The way to do this is a Ph.D topic in itself.

- A step ahead can be an investigation regarding other optimization algorithms to address future heuristics combining scheduling issues with current graph resolution.

Among the optimization techniques, we implemented only the ES and SA techniques. Other techniques e.g. GA which is also a very promising technique can to be implemented.

- More and more complete graphs including more and more standards can be drawn. We only dealt with PHY layer however the aim is to draw graph from application to PHY layer including intermediate layers to make cross-layer optimizations.
- A related research topic is to investigate other techniques to formulate cost function. We used the weighted sum approach to formulate our cost function. However there are other techniques that can be probed.
- An important continuation of this work is to find common operators: already existing ones or invent new ones and devise ways to use these operators to their utmost potential.
- An identified problem with costs needs to be further explored especially in the case of mixed designs consisting of FPGA, DSP et. This problem has been partially addressed and requires further analysis. Considering heterogeneous targets such as FPGAs and DSPs is not done yet. If so, this approach could be a real opportunity to make automatic HW/SW partitioning.
- A desirable continuation of this work will be towards the new paradigm called cross-layer optimization that seeks for performance improvements between different layers. Wireless communications are the most interesting research target for cross layering, due to the inherent variability of the radio channel and the potential enhancements that other layers can attain from knowing information about its state. Regarding wireless networks with infrastructure-based support, CDMA-based systems and WLAN systems have been also a extensive research target for cross-layer design, focusing also in specific applications as multimedia transmissions [199, 200]. Several issues must be considered when undertaking cross-layer designs [201]. Cross-layer can play a key role to achieve the ultimate goals of SDR.
- Long term perspective is to provide an industrial tool for designers of future SDR and CR equipments'
  - that allows to optimize multi-standards SDR design and
  - that permit to choose to orientate the design towards high granularity or low granularity (depending on designers' company interest), so that the approach can be used as well in a microelectronics company building ASICs (more low level CO oriented) or a processor builder (more high level CO oriented).

  but many problems need to be solved for that as discussed above.

It is certainly interesting to follow the research directions mentioned above to further enhance the ideas proposed in this thesis. Each of above topics would require at least one if not several new Ph.D.

Appendix

Appendix A

SDR projects

This appendix lists various European and French SDR projects.

A.1 European SDR projects

In the context of the ACTS and IST, Europe has funded several projects related to SDR [202, 203]. The main projects are listed below:

CAST: Configurable radio with Advanced Software Technology

DRIVE: Dynamic Radio for IP services in Vehicular Environments

E^{2}R: End-to-End Reconfigurability

E^{2}R-II: End-to-End Reconfigurability phase-II

FIRST: Flexible Integrated Radio System Technology

FRAMES: Future Radio Wideband Multiple Access Systems

MEDIAN: Wireless Broadband CPN/LAN for Professional and Residential Multimedia Applications

MOBIVAS: Download Mobile Value Added Services Through Software Radio and Switching integrated Platform

NEWCOM: Network of Excellence in Wireless Communications

ORACLE: Opportunistic Radio Communications in Unlicensed Environments

PASTORAL: Platform and Software for Terminals Operationally Reconfigurable

RAISIN: RAINbow extenSion for trial INtegration

SCOUT: Smart User-centric Communication Environment

SODERA: Reconfigurable radio for Software Defined Radio for 3rd generation mobile terminals

SORT: Software Radio Technology

SUNBEAM: Smart Universal Beamforming

TRUST: Transparent Reconfigurable Ubiquitous Terminal

In addition to these projects, the European Commission has participated to several actions (conferences, meeting and workshops) related to the SDR: ACTS in 1998 and IST Mobile and Wireless Communication Summit'00 to Summit'07.

A.2 French SDR Projects

In this section we quote the main RNRT projects:

A3S: Adéquation Architecture/Application Système

IDROMel: Impact des Equipements Reconfigurables pour le Déploiement des Futures Réseaux Mobiles

MOPCOM: Modélisation et spécialisatiOn de Plates-formes et COmposants MDA

PETRUS: Plateforme d'Evaluation des Technologies Radio pour l'UMTS-TDD et des Services

PLATON: PLATeforme Ouverte pour les Nouvelles générations de communications mobiles

PLATONIS: PLATeforme de validaTiOn et d'expérimentatioN multI-protocoles et multi-Services

RHODOS: Réseau Hybride Ouvert pour le Déploiement des Services mobiles

Appendix B

Complexity evaluation of air interface standards

This appendix lists complexity evaluation of various air interface standards.

B.1 Complexity evaluation of IEEE 802.11a

B.2 Complexity evaluation of UMTS

Table B.1: Computational complexity of IEEE 802.11a transmitter tasks

TASKS	**Computational Complexity of Data Processing Tasks**						
		Logical Oprs.	**Shift Reg. Accesses**	**I/P Oprs.**	**O/P Oprs.**	**LUT Accesses**	**Miscellaneous Oprs.**
Scrambling	2 XOR 8 XOR		14 (bit level)	1 1-bit	1 1-bit o/p oprs per input bit		
Convolutional Encoding			19 (bit level)	1 1-bit	2 1-bit o/p oprs per input bit		
Rate Independent Puncturing P1		13 (bit level)	13 (bit level)	13 1-bit	12 1-bit o/p oprs per input bit		
Rate Dependent Puncturing P2	$K = 3 for$ $r = 3/4$	3(bit level)	3(bit level)	3 1-bit	2 1-bit o/p oprs per 3 input bit		
	$K = 9 for$ $r = 9/16$	9 (bit level)	9 (bit level)	9 1-bit	8 1-bit o/p oprs per 9 input bit		
Interleaving (N_{CBPS} worst case value 288				N_{CBPS} 1-bit	N_{CBPS} 1-bit oprs per N_{CBPS} bit	$2xN_{CBPS}$ LUT ($N_{CBPS}x9$ bits)	
Constellation Map	BPSK			1 1-bit	2 o/p oprs per per 1 input bit	2 LUT (2 entries x 1 bit	
	QPSK			2 1-bit	2 o/p oprs per per 2 input bits	2 LUT (2 entries x 2 bits	
	16-QAM			4 1-bit	2 o/p oprs per per 4 input bits	2 LUT (2 entries x 4 bits	
	64-QAM			6 1-bit	2 o/p oprs per per 6 input bits	2 LUT (2 entries x 6 bits	
Pilot Insertion				2x48 word level	2x64 word level		
64 Point Complex IFFT (radix 4 DIF, in place with bit reversal implementation				2x168 word level 2x84 coeff. accesses and transfers			Multiplications: 336 Additions: 336 Subtractions: 336 per 64 i/p C samples
		384 AND		2x256 word level		768 LUT Oprs.	Additions:384 per 64 i/p C samples (bit r.) 2x64 div with Kmod per 64 i/p C samples (normalizations)
Cyclic Prefix Insertion				2x64 word level	2x80 word level		

Table B.2: Computational complexity of IEEE 802.11a receiver tasks

TASKS	**Computational Complexity of Data Processing Tasks**						
		Logical Oprs.	**Shift Reg. Accesses**	**I/P Oprs.**	**O/P Oprs.**	**LUT Accesses**	**Miscellaneous Oprs.**
Cyclic Prefix Extraction				2x64 w. level	2x64 w. level per 64 i/p c. samples		
Frequency Error Correction				2x64 w. level	2x64 word level	768 LUT Operations	
FFT				2x168 w. level 2x84 coeff. acc. & t/fs			Multp.: 336 Add.: 336 Subt: 336 per 64 i/p C. samples
		384 AND		2x256 w. level		768 LUT Operations	Add.:384 per 64 i/p C samples
Frequency Domain Equalization				2x48 w. level	2x48 w. level		Multp.:6x48 Add.:2x48 per 48 i/p c. samples
Constellation Demapping	BPSK			2 word level	1 1-bit o/p opr per i/p c. sample		
	QPSK			2 word level	2 2-bit o/p oprs per i/p c. sample	2(6 entries) 2(2 entries) 1-bit	Multp.:2 Add.:6
	16-QAM			2 word level	4 4-bit o/p oprs per i/p c. sample	2(6 entries) 4(2 entries) 4-bit	Multp.:2 Add.:6
	64-QAM			2 word level	6 8-bit o/p oprs per i/p c. sample	2(6 entries) 6(3 entries) 6-bit	Multip.:2 Add.:6
Deinterleaving				N_{CBPS} 1-bit	N_{CBPS} 1-bit o/p oprs per N_{CBPS}	2xN_{CBPS} (N_{CBPS} entries x 9-bits)	
Rate Dependent Puncturing P2	$K = 3 for$ $r = 3/4$	3 (bit l.)	3 (bit l.)	2 1-bit	3 1-bit o/p oprs per 2 input bit		Counter Incr.:3 Compare:3
	$K = 9 for$ $r = 9/16$	9 (bit l.)	9 (bit l.)	8 1-bit	9 1-bit o/p oprs per 8 input bit		Counter Incr.:3 Compr.:3
Rate Independent Puncturing P1		13 (bit l.)	13 (bit l.)	12 1-bit	13 1-bit o/p oprs per 12 i/p bits		Counter Incr.:K Compr.: K
Viterbi Decoding				2 1-bit	1 1-bit o/p oprs per i/p bit		Add.:$2^K = 128$ Compr.:$2^{K-1} = 64$
Descrambling		2 XOR	14 (bit l.)	1 1-bit	1 1-bit o/p oprs per i/p bit		

Table B.3: Computational complexity of UMTS, FDD UL Tx Blocks (CCTrCH and TrCH decoding, 1 Frame)

Function	**Additions and Subtractions**			**Multiplications**		**Division and Modulo**		**Total**
	Inc/Dec	Const (any)	Other (not const)	Const (2^n)	Const (any)	Const (2^n)	Const (any)	
CRC Encoder	3907	0	1	3865	0	30939	0	38712
Turbo Encoder (DTCH)	15447	0	0	7712	0	38578	0	61737
Convolutional Encoder	245	0	0	30	0	390	0	665
Radio Frame Equalization	11670	0	0	1	0	1	0	11672
1^{st} Interleaver	46710	0	11715	0	1	1	0	58427
Radio Frame Segmentation	11670	0	0	0	0	1	0	11671
Rate Matching	48575	7720	30971	2	2	2	6	87277
TrChMux)	115216	0	2	0	0	0	0	115218
Ph Channel Segmentation	9600	0	0	0	0	0	0	9600
2^{nd} Interleaver	48062	19200	1	0	1	0	0	67264
Ph Channel Mapping	96864	0	45	1	15	0	1	96926
Total	407964	26920	42735	11611	19	69913	7	559169

Table B.4: Computational complexity of UMTS, FDD DL Rx Blocks (CCTrCH and TrCH decoding, 1 Frame)

Function	**Additions and Subtractions**			**Multiplications**		**Division and Modulo**		**Total**
	Inc/Dec	Const (any)	Other (not const)	Const (2^n)	Const (any)	Const (2^n)	Const (any)	
PhChannel DeMapping	48099	0	45	0	0	0	0	48144
2^{nd} DeInterleaver	45661	0	18240	1	0	0	0	63903
2^{nd} DTX Remove	9120	0	0	0	0	0	0	9120
TrChDeMux	27369	0	0	2	0	2	4	27337
1^{st} DeInterleaver	54635	0	0	9156	1	0	1	63793
1^{st} DTX Remove	9120	0	0	1	1	0	0	9122
Rate DeMatching	44169	0	0	10360	0	0	0	54529
Viterbi Decoder)	53946	30	30	92190	0	0	92190	253777
Turbo Decoder (DTCH)	5576579	0	493953	4198612	0	0	987904	12985888
Ph Channel Mapping	3907	0	0	1	0	0	30939	38712
Total	5872604	30	512268	4310323	3	2	1111038	13554364

Appendix C

Optimization tool

C.1 Introduction

We developed our optimization tool using C++. The *brute force* and *simulated annealing* techniques of optimization have been implemented. This tool has four major parts that correspond to the graph models of SDR equipments and their solutions i.e. Nodes (class Hnode), Arcs (class Harc), Graph (class Hgraph) and Optimization (class Optimization). In the following we list some of important methods that belong to these four classes. The names of the methods and their short description is provided below.

C.1.1 Methods of nodes

1. `void Hnode::change_etat();` Changes the state of the node (select/de-select)
2. `void Hnode::set_name(string id);` Gives a name to a node for its identification.
3. `void Hnode::set_cout(double c);` Sets the basic building cost of a node.
4. `void Hnode::set_cout_base(double c);` Sets the initial computational cost of node.
5. `void Hnode::set_type(int t);` Sets the type of node e.g. FPGA/DSP.

C.1.2 Methods of arcs

1. `void Harc::add_arc(pHarc ha);` Adds an outgoing arc.
2. `void Harc::add_remonte(pHarc ra);` Adds an incoming arc.
3. `bool Harc::is_or() const;` Tests that a hyperarc is an OR hyperarc.
4. `bool Harc::is_and() const;` Tests that a hyperarc is an AND hyperarc.

C.1.3 Methods of graph

1. `void Hgraph::add_arc(pHnode ori, pHnode dest, double poids, double dist);` Adds an OR hyperarc to the graph.

2. `void Hgraph::add_Harc(pHnode ori, double dist, vector<pHnode> dests, vector<double> pds);` Adds an AND hyperarc to the graph.
3. `pHnode Hgraph::add_node(string name);` Adds a useless node to the graph. A node is said to be useless if the cost values are not assigned to it.
4. `pHnode Hgraph::add_node(string name, double cout, double c_fpga);` Adds a useful node to the graph.
5. `void Hgraph::trier_nodes();` Create vectors of useless/useful nodes.
6. `bool Hgraph::verif_arcs();` Verifies that all the arcs in the graphs are valid.
7. `void Hgraph::affiche_pris();` Shows the nodes that are selected in a particular solution.
8. `void Hgraph::affiche_solution();` Shows the nodes and intermediate execution cost associated with each node.
9. `double Hgraph::calcule_cout();` Calculates the cost of the solution in global. It is an intelligent function developed to calculate cost using different sets of useful nodes.

C.1.4 Methods of optimization

1. `bool Optimisation::pre_verif();` Verifies that the problem has a solution or not.
2. `void Optimisation::brute_force();` Implements brute force method of optimization for determining the solutions.
3. void `Optimisation::recuit_simple(int temperature);` Implements simulated annealing method of optimization to determine the solutions to the problem.
4. `void Optimisation::init_recuit();` Initializes the simulated annealing algorithm.
5. `void Optimisation::recuit_rec(int temperature_max, int temperature);` The function of decreasing temperature (in simple SA algorithm) is a linear function. Advantage of using a linear function is simplicity of implementation but on the other hand we have some decrease in the effectiveness of SA for optimization.
6. `void Optimisation::recuit_rec2(int temperature_max, int temperature,double Kk);` The is another version of SA algorithm. The function of decreasing temperature in this case is an exponential function.
7. `void Optimisation::recuit_mixte(bool glout, int gc, int temperature);` This function launches the simulated annealing algorithm with some heuristics.

By applying above methods/functions of our tool to the graph models we can find the optimum solution that balances between building and computational costs. Details of the implementation of these methods are deliberately left because of space limitations.

Appendix D

Channel coding and FFT

D.1 The Fourier transform over finite fields

Transforms over Galois field have been introduced into the study of error control codes in order to reduce decoder complexity first by Gore [204], later by Michelson [205], Lempel and Winograd [206] and Chien [207]. Blahut [208], showed that the idea of coding theory can be described in a frequency domain setting that is much different from the familiar time domain setting. In this context, by frequency domain reasoning, he described several encoder and decoders schemes. According to Pollard [187], the Fourier transform in a Galois field closely mimics the Fourier transform in the complex field with minor difference.

D.1.1 Frequency encoding of RS codes over $GF(2^m)$

Let us consider the RS codes defined over $GF(2^m)$.
Definition 2.3.3. *A RS code of block length n over $GF(q)$, with n a divisor of $q-1$, is defined as the set of all words over $GF(q)$ of length n whose Fourier transform is equal to zero in a specified block of $d-1$ consecutive components, denoted $\{j_0, j_0+1, ..., j_0+d-2\}$.*
The block-length of an RS code over $GF(q)$ is directly related to the order of α. If α has an order $n = q-1$, α is a primitive element and the RS code is called a *primitive RS code*. The minimum distance of an RS code is:

$$d_{min} = n - k + 1 \tag{D.1}$$

where k is the block-length of the information sequence.
As for any cyclic code, the encoder of RS codes can be either systematic or non systematic. One may encode in the natural way using the generator polynomial. This encoder is called a *time domain encoder*. Alternatively, one may choose to encode the RS codes directly in the transform domain by using the data symbols to specify spectral components. This is called a *frequency domain encoder*. Frequency encoding of an $RS(n,k)$ code is as follows. Some set of $d-1$ frequencies, indexed by $j = j_0, j_1, ..., j_0 + d - 2$, is chosen as the set of spectral components constrained to zero. The $n - d + 1$ unconstrained components of the spectrum are filled with data symbols from $GF(q)$. A non systematic codeword is produced by taking the inverse Fourier transform of the frequency vector. The obtained RS code is referred to as $RS(n,k)$ where $k = n - d + 1$. The common choice for j_0 is $j_0 = 1$ and the corresponding generator polynomial is:

$$\mathbf{g}(x) = (x-\alpha)(x-\alpha^2)...(x-\alpha^{2t}). \tag{D.2}$$

This is always a polynomial of degree $2t$. As an example, let us consider the code RS(7,5) with a polynomial generator

$$g(x) = (x-\alpha)(x-\alpha^2) = x^2 + \alpha^4 x + \alpha^3 \tag{D.3}$$

and an information sequence

$$\mathbf{m}(x) = \alpha^6\ x^4 + x^3 + \alpha^4 x^2 + \alpha\ x + 1. \tag{D.4}$$

Using the two encoding techniques, we shall find the two codewords corresponding to $\mathbf{m}(x)$.

1. Frequency domain encoding: The components that should be constrained to zero are the components indexed by 1 and 2. Thus, the spectral vector can be written as

$$\mathbf{C} = [1\ \ 0\ \ 0\ \ \alpha\ \ \alpha^4\ \ 1\ \ \alpha^6] \tag{D.5}$$

and the non systematic time domain codeword

$$\mathbf{c} = IFFT(\mathbf{C}) = [1\ \ \alpha\ \ \alpha\ \ 0\ \ \alpha^5\ \ \alpha\ \ \alpha^6] \tag{D.6}$$

2. Time domain encoding: The time domain encoding consists in multiplying $m(x)$ by $x^{n-k} = x^2$ and performing the polynomial division $x^2\ \mathbf{m}(x)$ by $g(x)$; The coefficients of the remainder $v(x)$ of this division represent the redundancy symbols. The resultant systematic codeword

$$\mathbf{c} = [\alpha^3\ \ \alpha^2\ \ 1\ \ \alpha\ \ \alpha^4\ \ 1\ \ \alpha^6] \tag{D.7}$$

The time domain encoding can be realized with a division circuit which is a linear $(n-k)$ stage shift registers with feedback connection based on the generator polynomial $\mathbf{g}(x)$.

D.1.2 Frequency decoding of RS codes over $GF(2^m)$

The received codeword $\mathbf{r}$ can be written as the sum $\mathbf{r} = \mathbf{c} + \mathbf{e}$. The decoder must process the received codeword $\mathbf{r}$ so that the error word $\mathbf{e}$ is removed to obtain the transmitted codeword $\mathbf{c}$. The Fourier transform of the received word has components $R_j = C_j + E_j$ for $j = 0, 1, ..., n-1$. The $2t$ spectral components, from j_0 to $j_0 + 2t - 1$, are called the (frequency domain) syndromes. The syndromes can be written as

$$S_j = R_{j+j_0-1} = E_{j+j_0-1} \qquad j = 1, ..., 2t. \tag{D.8}$$

It is convenient to index the syndromes starting with index one. To simplify this equation, we take $j_0 = 1$ and therefore $S_j = R_j$. The block of $2t$ syndromes provides the window through which the decoder can start the decoding process to recover the data. Let us consider the error vector with its polynomial form

$$e(x) = e_{n-1}x^{n-1} + e_{n-2}x^{n-2} + ... + e_1x + e_0. \tag{D.9}$$

The $2t$ syndromes can be defined by

$$S_j = \mathbf{r}(\alpha^j) = \mathbf{c}(\alpha^j) + \mathbf{e}(\alpha^j) = \mathbf{e}(\alpha^j) \qquad j = 1, ..., 2t. \tag{D.10}$$

As the code can correct at most t errors, the error polynomial has at most t coefficients that are nonzero. Suppose that t_1 errors occur, $0 \leq t_1 \leq t$ and that they occur at the unknown locations $i_1, i_2, ..., i_{t_1}$. The error polynomial can be written as

$$e(x) = e_{i_{t_1}}x^{i_{t_1}} + ... + e_{i_2}x^{i_2} + e_{i_1}x^{i_1} \tag{D.11}$$

where e_{i_l} is the magnitude of the lth error. Then, the syndrome S_1 evaluated at α can be written as

$$S_1 = e(\alpha) = e_{i_{t_1}}\alpha^{i_{t_1}} + ... + e_{i_2}\alpha^{i_2} + e_{i_1}\alpha^{i_1}. \tag{D.12}$$

Similarly, we can evaluate the other $2t-1$ syndromes at $\alpha^2, ..., \alpha^{2t}$. For simplicity of notation, the error values e_{i_l} are denoted by Y_l and the error location numbers α^{i_l} by X_l. Now, we can write the $2t$ as a set of simultaneous equations:

$$S_1 = Y_{t_1}X_{t_1} + Y_{t_1-1}X_{t_1-1} + ... + Y_1X_1 \tag{D.13}$$

$$S_2 = Y_{t_1}X_{t_1}^2 + Y_{t_1-1}X_{t_1-1}^2 + ... + Y_1X_1^2 \tag{D.14}$$

$$\vdots$$

$$S_{2t} = Y_{t_1}X_{t_1}^{2t} + Y_{t_1-1}X_{t_1-1}^{2t} + ... + Y_1X_1^{2t} \tag{D.15}$$

The decoding problem has now been reduced to the problem of solving a system of nonlinear equations. This set of equations is too difficult to solve directly. Instead, a polynomial called locater polynomial is introduced. This polynomial that has t_1 nonzero coefficients is given by

$$\Lambda(x) = \Lambda_{t_1}x^{t_1} + \Lambda_{t_1-1}x^{t_1-1} + ... + \Lambda_1x + 1 \tag{D.16}$$

and defined to be the polynomial with zeros at the inverse error locations X_l^{-1} for $l = 1, ..., t_1$. That is,

$$\Lambda(x) = (1 - xX_1)(1 - xX_2)...(1 - xX_{t_1}). \tag{D.17}$$

The task now is to find the coefficients of $\Lambda(x)$. Once the coefficients are known, the error locations can be obtained by computing the zeros of $\Lambda(x)$. After manipulating the equations (D.16) and (D.17) we finally reach at the following equation

$$S_{j+t_1} = -(\Lambda_{t_1}S_j + \Lambda_{t_1-1}S_{j+1} + ... + \Lambda_1S_{j+t_1-1}), \tag{D.18}$$

for $1 \leq j \leq 2t - t_1$. Equation D.18 can be written in general form of linear recursion

$$S_k = -\sum_{j=1}^{t_1} \Lambda_j S_{k-j} \ mod\ n \qquad k = t_1 + 1, ..., 2t_1. \tag{D.19}$$

Clearly, if we consider the more general case where t errors have occurred, the equation (D.19) becomes

$$S_k = -\sum_{j=1}^{t} \Lambda_j S_{k-j} \ mod\ n \qquad k = t + 1, ..., 2t. \tag{D.20}$$

Obviously, the number of errors in the received word is not known. The principle of decoding is to solve the linear recursion for Λ using the smallest value of t_1 as pointed out by Blahut in [209]. This system of equations is always solvable for Λ and t_1. To solve it there is two ways: direct method and iterative method. In both methods, the decoding process consists of two principal steps: finding the error locations and finding their magnitudes.

Direct method

To find the error locations, the linear recursion of equation (D.20) can be solved by an algorithm proposed by Peterson [210]. To find the error magnitudes, the Gorenstein-Zierler algorithm [211] is applied. The bottleneck here is that Petreson's algorithm involves matrix inversion which is reasonable only for small values of t. Since number of multiplications necessary to invert a t by t matrix is proportional to t^3. However, when t is large, one should use a more efficient method that obviously will be more intricate but computationally much simpler. The solution is the iterative method.

Iterative method

The key step in the RS decoding process is the computation of the error locater polynomial $\Lambda(x)$ from the known $2t$ spectral components of the received words. Berlekamp [212] developed an elegant algorithm based on the language of polynomials which has been described in terms of shift registers by Massey [213]. This algorithm known as *Berlekamp-Massey algorithm* solves the problem by an iterative procedure that consists in designing the shortest linear recursion $(\Lambda(x), t_1)$ capable to produce the known sequence of syndromes. The algorithm consists of $2t$ iterations of computation. At each iteration, a candidate syndrome $\bar{S_k}$ is computed

$$\bar{S_k} = -\sum_{j=1}^{L_k - 1} \Lambda_j^{k-1} S_{k-j} \qquad k = 1, ..., 2t, \tag{D.21}$$

where L_k is the length of linear recursion. The obtained syndrome $\bar{S_k}$ is subtracted from the desired (or known) syndrome S_k to get a quantity

$$\Delta_k = S_k - \bar{S_k} = S_k - \sum_{j=1}^{L_k - 1} \Lambda_j^{k-1} S_{k-j}. \tag{D.22}$$

If Δ_k is zero, then the $(\Lambda(x), L_k)$ is the right linear recursion. Otherwise, $(\Lambda(x), L_k)$ should be modified. The main trick of Berlekamp-Massey algorithm is to use earlier iterations to compute a new minimum-length linear recursion $(\Lambda(x), L_k)$.

Berlekamp-Massey algorithm. In any field, let $S_1, S_2, ..., S_{2t}$ be given. With initial conditions $\Lambda^{(0)}(x) = 1$, $B^{(0)}(x) = 1$, and $L_0 = 0$, let the following set of equations for $k = 1, ..., 2t$ be used iteratively to compute $\Lambda^{(2t)}(x)$:

$$\begin{aligned} \Delta_k &= \sum_{j=0}^{n-1} \Lambda_j^{k-1} S_{k-j}, \\ L_k &= \delta_k(k - L_{k-1}) + (1 - \delta_k)L_{k-1}, \\ \Lambda^{(k)}(x) &= \Lambda^{(k-1)}(x) - \Delta_k x B^{k-1}(x), \\ B^{(k)}(x) &= \Delta_k^{-1}\delta_k \Lambda^{(k-1)}(x) + (1 - \delta_k) x B^{k-1}(x) \end{aligned} \tag{D.23}$$

where $\delta_k = 1$ if both $\Delta_k \neq 0$ and $2L_{k-1} \leq k - 1$, and otherwise $\delta_k = 0$. Then $(\Lambda^{2t}(x), L_{2t})$ is a linear recursion of shortest length that produces $S_1, S_2, ..., S_{2t}$. Once the error locater polynomial is computed, the next step is to compute the error values. There are two approaches to perform this last step.

The first approach is to continue the iterative computation. That is, after the $2t$ iterations of computation of $\Lambda(x)$ that produce the $2t$ syndromes, the remaining $n - 2t$ syndromes are produced by computing the discrepancy Δ_k for $k = 2t + 1$, ..., n and at each iteration compute $S_k = S_k - \Delta$. Fig. D.1 shows a flow diagram for a decoder that computes the frequency-domain codeword $\mathbf{C}$ from the frequency received word $\mathbf{R}$.

The second approach is based on two elegant algorithms: Chien algorithm [214] to find the error locations and Forney algorithm [215] to compute the error values. Chien algorithm searches iteratively the roots of polynomial $\Lambda(x)$ that can specify the error locations. This algorithm tests the condition

$$\sum_{j=1}^{t} \Lambda_j \alpha^{ij} = 1 \qquad i = 1, 2, ..., n - 1. \tag{D.24}$$

If this sum is 1, then the $(n - i)th$ component of the received word is erroneous. By rewriting equation D.24 as

$$\sum_{j=0}^{t} \Lambda_j \alpha^{ij} = 0 \qquad i = 1, 2, ..., n - 1, \tag{D.25}$$

where $\Lambda_0 = 1$. In fact, the n coefficients of equation D.25 represent the frequency components of polynomial $\Lambda(x)$. Then, by computing the FFT of $\Lambda(x)$, the error locations that correspond to the roots of $\Lambda(x)$ can be determined by testing the spectral components $\Lambda(x)$. That is, if $\Gamma_j = \sum_{j=0}^{t} \Lambda_j \alpha^{ij} = 0$, the $(n - i)th$ symbol is erroneous. This frequency reasoning is advantageous if an efficient algorithm to compute the $FFT(\Lambda(x))$ can be applied. Then, the important task of RS decoding *Chien search* can be performed with FFT.

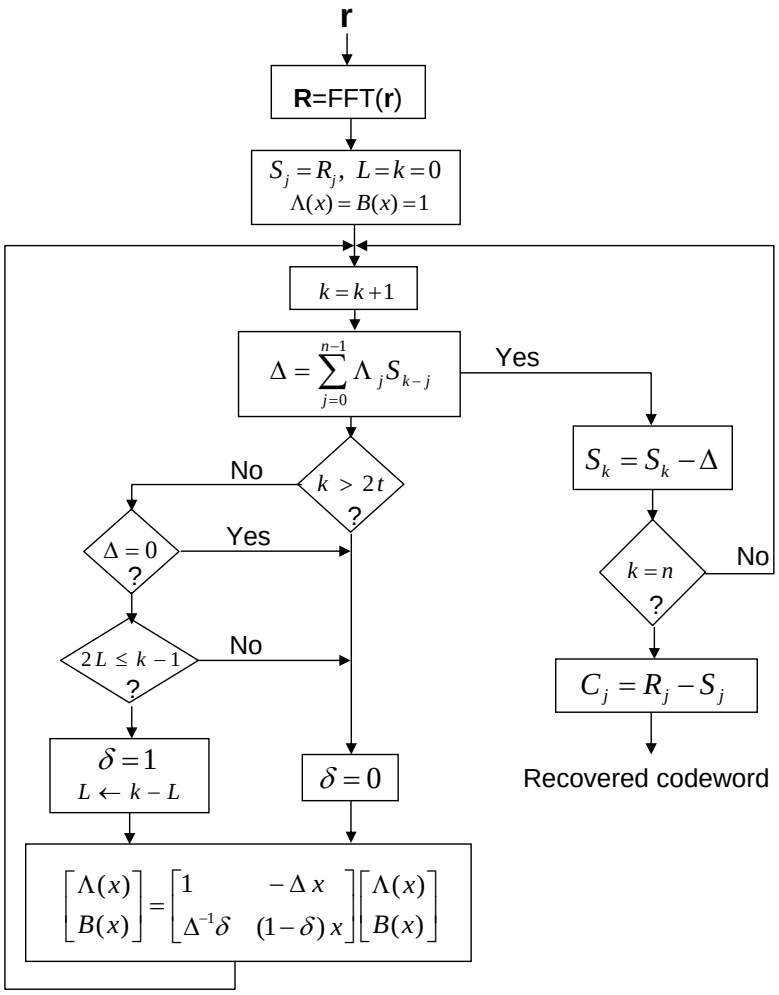

Figure D.1: A frequency-domain decoder

The error values can be computed using Forney algorithm which evaluates a polynomial called *evaluator polynomial* ($\Omega(x)$) at the specified error locations. The polynomial $\Omega(x)$ can be computed using the polynomial multiplication

$$\Omega(x) = \Lambda(x)S(x) \quad mod(x^{2t+1}), \tag{D.26}$$

where

$$S(x) = \sum_{j=1}^{2t} S_j x^j.$$

The flow diagram of Fig. D.2 shows the frequency decoder using Forney algorithm where $\Lambda'(x)$ is the derivative of $\Lambda(x)$. More details about the algorithm can be found in [209].

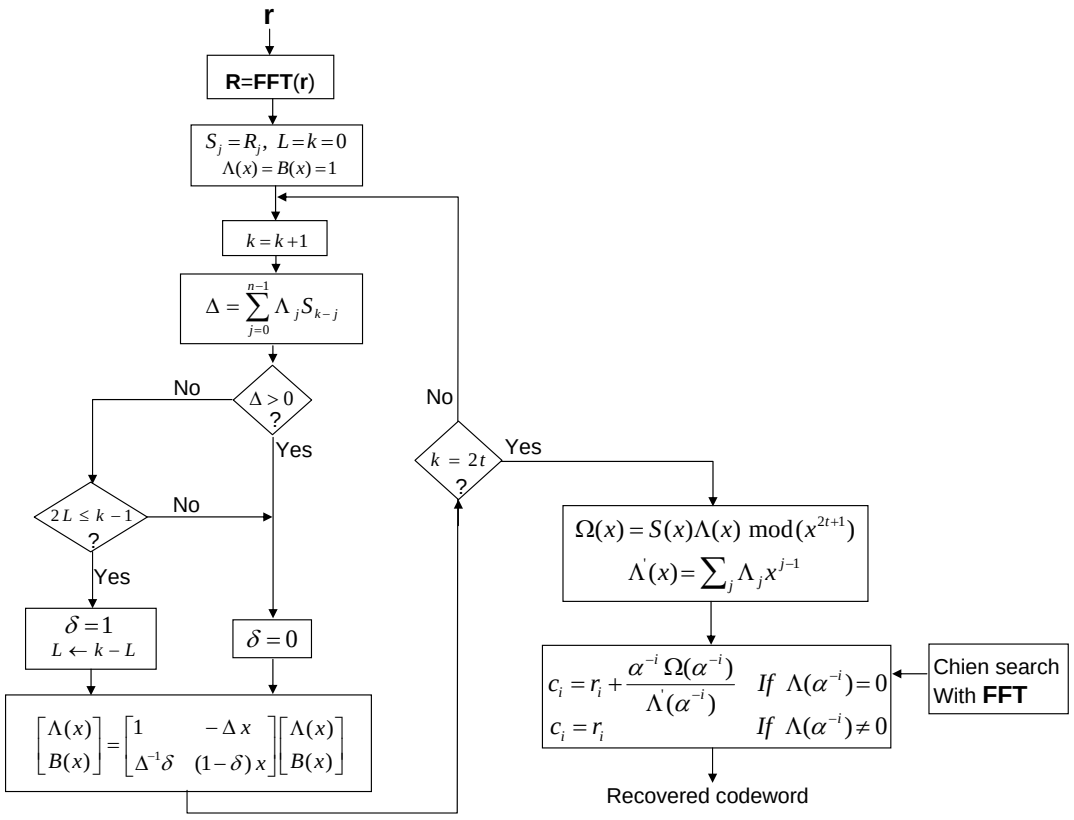

Figure D.2: A frequency-domain decoder using the Forney algorithm

We note that there are other important algorithms known in the literature used in the decoding of RS codes. Among these algorithms, we find Sugiyama, Euclide, Gao and other algorithms. The study of these various algorithms is not the subject of our work. The goal is to highlight the role of the FFT in the encoding and decoding processes by reviewing the most important algorithms adapted to the frequency processing of RS codes.

D.1.3 Comparison between time and frequency domain decoding of RS codes

In this section we give a comparison between frequency domain and time domain decoding of RS codes. All the notations for time domain decoding and details of time domain RS decoder can be found in [216]. For the frequency domain approach, the matrix update, i.e. update of $\Lambda(x)$ and $B(x)$, requires at most $2t$ multiplications per iteration and the calculation of Δ_k requires at most t multiplications per iteration. There are $2t$ iterations and so no more than $6t^2$ multiplications are required to compute the error locater polynomial. In the other hand, the time domain approach works directly on the raw data word as received without any transform. The frequency-domain vectors of Berlekamp-Massey algorithm of length t are replaced by time-domain vectors of length n. Then the time domain decoder has a computation complexity proportional to n^2 while the one of the frequency domain

decoder is proportional to t^2.

The time domain decoder can be attractive because there is no syndrome computation or Chien search. This means that the decoder has a very simple structure, but the penalty is a longer running time. As for the frequency domain decoder, its structure is more complex but this drawback is largely compensated by its major advantage that is a shorter running time. For this latest reason, the frequency domain decoders are more attractive and have found their use in various applications and specifically in applications that require high throughput rates.

D.2 Fermat Transform Based RS Codes

In this section we describe RS codes defined over $GF(F_t)$ and we show that these codes have almost the same performances in terms of Bit Error Rate (BER) and Frame Error Rate (FER) compared to the classical RS codes defined over $GF(2^m)$. We are interested in RS codes defined over $GF(F_t)$ because of the fact that their hardware implementation can be partially performed with the aid of pre-implemented FFT where the building of a SR system is considered.

D.2.1 Encoding of RS codes defined over $GF(F_t)$

The principles of RS encoding as described in section D.1.1, can be applied to the RS codes defined over $GF(F_t)$ with the only difference that the arithmetic operations are done modulo F_t rather than modulo 2. The encoding in the time domain uses a generator polynomial to calculate the parity check symbols. In the frequency domain, the encoding consists in constraining some specified spectral components to zero and filling the unconstrained components of the spectrum with data symbols of $GF(F_t)$. Then, the inverse Fermat transform generates a non systematic codeword.

D.2.2 Decoding of RS codes defined over $GF(F_t)$

Different algorithms used for the decoding of RS codes defined over $GF(2^m)$ as explained in section D.1.2 can be applied to decode the RS codes defined over $GF(F_t)$. However, the highly used RS decoding procedure in actual applications is that composed of the Berlekamp-Massey algorithm, Chien algorithm and Forney algorithm (see Fig. D.2).

D.2.3 Performances comparison between RS over $GF(F_t)$ and RS over $GF(2^m)$

In this subsection, we give simulation results in terms of BER and FER vs $\frac{E_b}{N_0}$ (E_b is the energy per bit and N_0 the power spectral density of the channel noise) for different RS coding schemes with correcting capacity t_c equal to 2. In this simulation, random data with a BPSK modulation are transmitted over an Additive White Gaussian Noise (AWGN) channel. In Fig. D.3 and Fig. D.4, we have a set of five BER and FER curves: (i) uncoded; (ii) RS(16,12) over $GF(17)$ with frequency encoding and frequency decoding using the Berlekamp-Massey algorithm and the recursive extension, the codeword symbols

are represented by 5 bits each; (iii) RS(16,12) over $GF(17)$ with time domain encoding and frequency domain decoding using the algorithms indicated in Figure 6.4, the information symbols and the parity check symbols are represented by 4 bits (4-4); (iv) it is the same case of (iii) with the only difference that each parity check symbol is transmitted over 5 bits (5-4); (v) RS(15,11) over $GF(16)$.

We start this analysis with curve (ii) of Fig. D.3. In this case, for low $\frac{E_b}{N_0}$ values, the BER is high compared to other coding schemes. This phenomena can be explained by the following: firstly, the loss of useful data rate by transmitting each symbol over $2^t + 1$ bits increases the channel error probability and consequently the BER. Secondly, if the number of erroneous symbols is higher than the correcting capacity, the decoding fails and since the symbols of code words are generated by the Fermat transform, the symbols will be all erroneous resulting in an high BER. Curve (iii) also indicates a high BER due to the fact that the symbol $2^{2^t} = 16$ is set to zero knowing that its probability occurrence is not equal to zero. We have calculated this probability for different $\frac{E_b}{N_0}$ values to be around 5.8 %, which in turn degrades the code performances. This degradation is clearly illustrated in Fig. D.4 where we observe that at high (E_b/N_0), the performances in terms of FER of curve (iii) worsen as compared to curves (ii). This can be explain by the fact that the probability of occurrence of symbol 2^{2^t} (estimated to be around 5.8 %) and that does not depend on the energy of signal, deliberately decreases the code's error correcting capability.

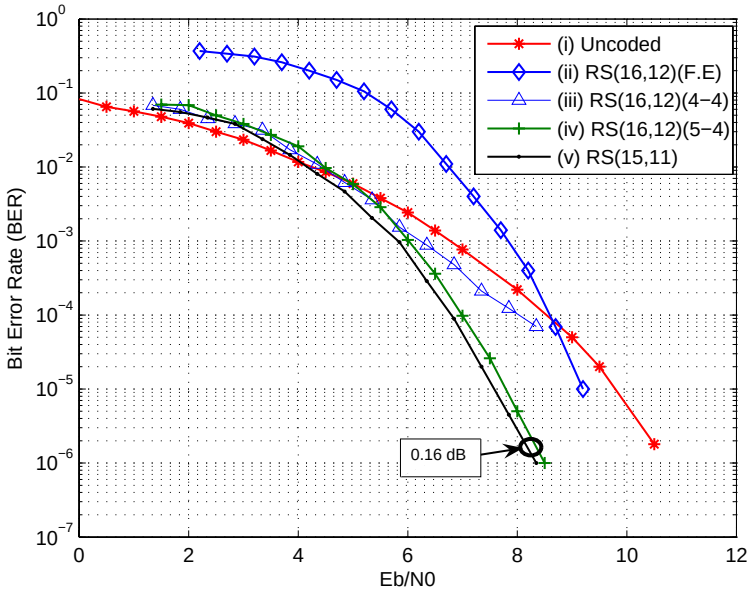

Figure D.3: Performances of RS(16,12) over $GF(17)$ and RS(15,11) over $GF(16)$ with BPSK modulation on AWGN channel. (F.E: Frequency Encoding))

The influence of this pre-committed error becomes more evident at high (E_b/N_0) where the RS codes represented by curve (ii) of Fig. D.4 begin to be more efficient.

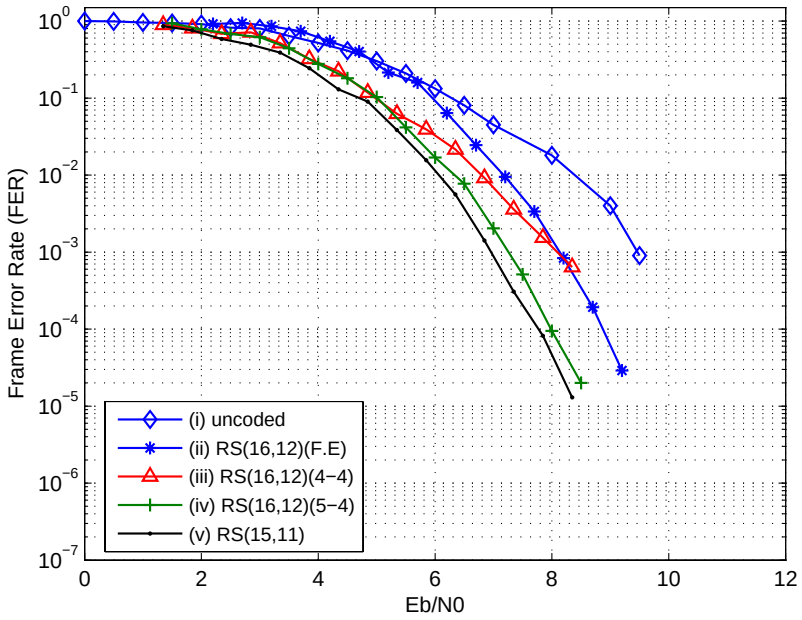

Figure D.4: FER of RS(16,12) over $GF(17)$ and RS(15,11) over $GF(16)$ with BPSK modulation on AWGN channel.

Now, let us consider the two curves (iv) and (v) of Fig. D.3. To solve the problem of the symbol "2^{2^t}", we have transmitted each information symbol over 2^t bits and each parity check symbol over $2^t + 1$ bits. In this way, the symbol "2^{2^t}" is correctly represented. Fig. D.3 shows that the performances of the RS(16,12) defined over $GF(17)$, using the time domain encoding and the frequency domain decoding according to the algorithms presented in Fig. 6.4, has similar performances as RS(15,11) defined over $GF(16)$. The asymptotic difference is 0.16 dB and can be expressed by the expression (in dB):

$$\Delta = 10log(1/R_1) - 10log(1/R_2), \tag{D.27}$$

where R_1 and R_2 are the code rates of RS(15,11) and RS(16,12) respectively. Here, $\Delta = 10log(15/11) - 10log(68/48) = 0.16$ dB. This difference decreases with the code length. To justify this, let us consider the two RS codes: RS(255,223) over GF(256) and RS(256,224) over GF(257), where the codes of each pair have the same error correcting capability (t_c). The associated code rates R_1 and R_2 are equal to 0.87 and 0.86 respectively. Applying equation D.27, Δ in this case is equal to 0.05 dB. This drop of Δ can be explained also by evaluating, for each RS code, the channel error probability

$$P_i = \frac{1}{2} erfc\sqrt{R_i \frac{Eb}{N_0}}\ , \quad i = 1, 2. \tag{D.28}$$

Fig. D.5 shows the channel error probability of each RS code. It is to be noticed that the difference between the two curves of channel error probability in the case of RS(255,223) and RS(256,224) is very low compared to that of the RS(15,11) and RS(16,12). Theoretically, the RS codes over $GF(F_t)$ have the same error correcting capability, the small difference in terms of BER is due to the difference between the channel error probabilities. In turn, the channel error probability varies according to the code rate. The lower the difference between the code rates, the lower the difference between the channel error probabilities. Consequently, for long RS codes the difference in term of BER becomes negligible.

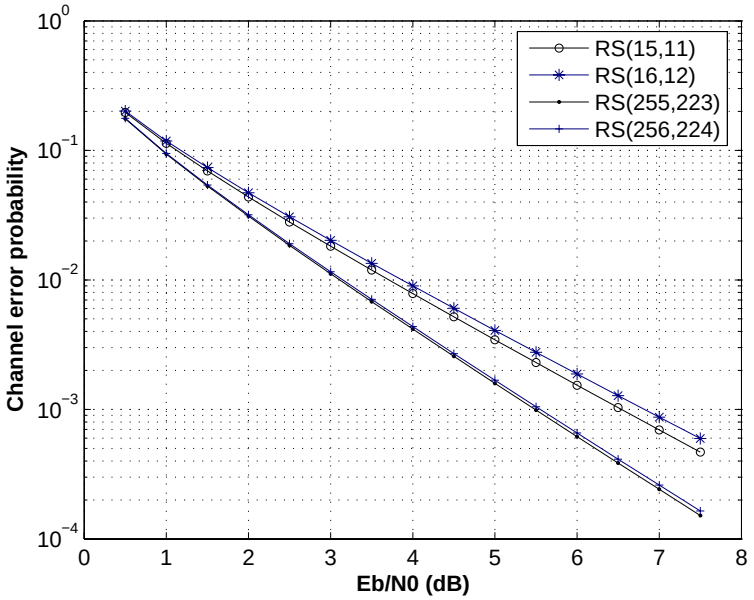

Figure D.5: The channel error probability for different RS codes.

D.3 DMFFT architecture

There are many different ways of implementing DMFFT operator. There can be two extreme architectures:

1. Perform the transform computations by using only a single memory unit and one RBPE. This method leads to a very simple circuit in terms of area, but the penalty is the very high computation time required to execute the transform.

2. Perform the transform computations by implementing the entire FFT structure composed of $log_2 N$ stages and $\frac{N}{2}$ RBPE units in each stage, where N is the transform length. For this method, no RAM blocks are needed between two consecutive stages since the entire sequence is processed in parallel. This method leads to a very high speed computation at a very large area.

We can find an architecture between two extremes. To compute transform length N, the strategy adopted is to implement $log_2 N$ computation stages where each stage is composed of one processing element (known as RBPE), a stage control unit and some memory blocks. With this implementation strategy, the DMFFT and the Velcro operators are implemented and compared. Based on this choice, let us consider the proposed block diagram of the DMFFT operator as shown in Fig. D.6. This block diagram contains a Global Control Unit (GCU) and $log_2 N$ stages.

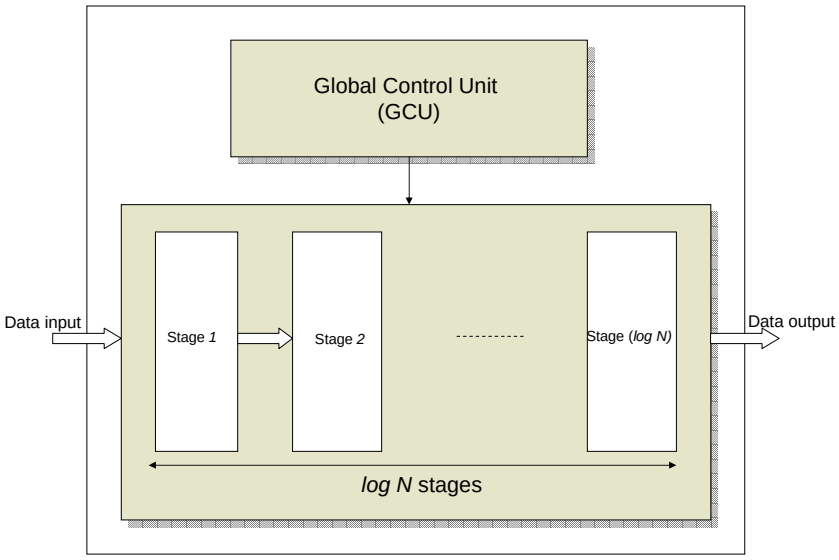

Figure D.6: The DMFFT architecture

The GCU is composed of the following individual circuits:

1. *FFT/FNT operation selection*: there is a control signal DM which determines the operating mode. The values 1, 0 of DM indicate that the complex Fourier transform and Fermat transform are performed respectively.

2. *Transform size selection*: the DMFFT operator is designed to perform various lengths of FFT and FNT computations. The parameter m determines the number of stages to be implemented and then the transform size $N = 2^m$.

3. *Word-length size*: to provide the system designer with maximum flexibility, the input/output data and the twiddle factor's word-length has been designed in a way to vary by a simple adjustment of corresponding parameters. A parameter n_c determines the complex word-length, n_w determines the twiddle factor word-length and t determines the desired Fermat number and subsequently the length $n = 2^t + 1$ of the $GF(F_t)$ valued symbols.

This GCU is also responsible for the initialization of the entire DMFFT architecture and for the control of the data input and data output and the synchronization between two consecutive input frames.

D.3.1 Stage architecture

The proposed FFT architecture is a pipeline architecture where a pipeline level is employed between two consecutive stages. The internal structure of each stage except stage 1 is illustrated in Fig. D.7. As shown, the stage architecture is composed of the following design units: SCU (Stage Control Unit), AGU (Address Generating Unit), memory blocks (RAM and ROM) and RBPE. As for stage 1, its architecture is similar to the one presented in Fig. D.7 with some modifications.

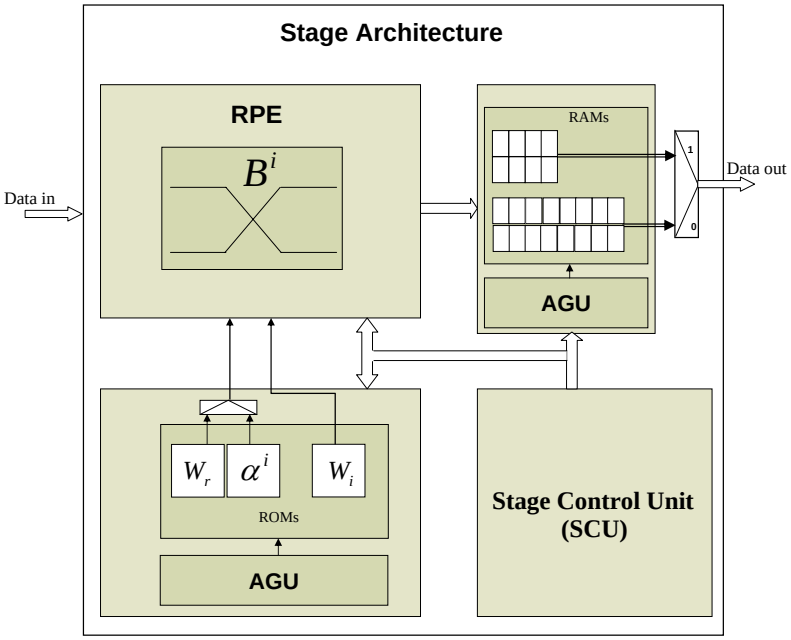

Figure D.7: The general architecture of a DMFFT stage

In this stage, no multipliers nor ROM blocks are needed since the corresponding twiddle factors are equal to 1. The RBPE as shown in Fig. D.8. represents the core of the DMFFT operator. RBPE is implemented in each stage and its operating mode is controlled by the GCU. The data stream acquisition of a RBPE employed in stage i is controlled by the SCU of stage $i-1$. The RBPE acquires the data and applies the multiplication of the twiddle factors with the corresponding symbols and the intermediate results continue the way to the adders and subtracters to undergo the corresponding operations. The storage of the RBPE computation results in the RAM blocks is handled by the SCU. Details of the internal blocks of RBPE can be found in [43].

D.3.2 RBPE complexity study

Table D.1 shows some measures of the implementation on Stratix II of the reconfigurable and Velcro butterflies for different n_c and n values that represent the $\mathbb{C}$ and $GF(F_t)$ symbol word-lengths respectively. The values $n = 9$ and $n = 17$ are the required bits number to map the symbols in $GF(F_t = 2^8 + 1 = 257)$ and $GF(F_t = 2^{16} + 1 = 65537)$ respectively.

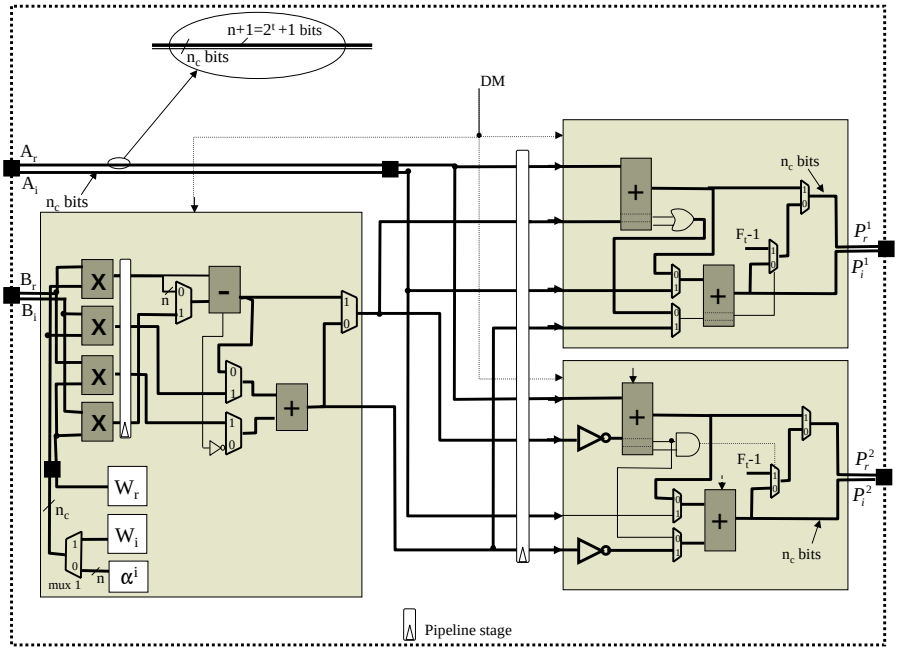

Figure D.8: The reconfigurable butterfly architecture

Table D.1: Comparison between the reconfigurable butterfly and the Velcro butterflies on a STRATIX II, EP2S15F484C3 device.

circuit	n_c=9 n=9	n_c=12 n=9	n_c=17 n=17
Velcro butterfly	403 ALUTs 4.20 ns	629 ALUTs 5.07 ns	1062 ALUTs 5.60 ns
Re-configurable butterfly	326 ALUTs 4.3 ns	514 ALUTs 5.18 ns	875 ALUTs 5.64 ns
ALUT's gain	19.1 %	18.2 %	17.6 %
Delay's excess	2.18 %	2.16 %	0.7 %
$\eta = \frac{1}{TC} * 10^6$	$\eta_V = 590$ $\eta_R = 713$	$\eta_V = 313$ $\eta_R = 375$	$\eta_V = 168$ $\eta_R = 202$
Performance-to -cost ratio gain	**20.8**%	**19.8**%	**20.2**%

Now let us discuss the figures given in Table D.1. For the reconfigurable butterfly, there is a small delay excess of around 2% compared to the Velcro solution. This time penalty is widely balanced by complexity gain values (in term of Adaptive Lookup Tables (ALUTs)) around 18% in favor of the common butterfly operator. Moreover, the performance-to-cost

ratio η_R has always the highest value compared to η_V. The associated gains are 20.8, 19.8 and 20.2 % for n_c equal to 9, 12 and 17 respectively. These performance figures clearly justify once again the interest of using a common and reconfigurable operator instead of *Velcro* operator.

D.4 Altera's Stratix-II FPGA family

This section gives a brief overview of useful features of Altera's Stratix-II FPGA family. Stratix II presents an efficient logic structure that maximizes the performance and enables device densities approaching 180,000 equivalent Logic Elements (LEs). Stratix II devices offer up to 9 Mbits of on chip, TriMatrix memory and has up to 96 DSP (Digital Signal Processing) blocks with up to 384 ($18-bit \times 18-bit$) multipliers for efficient implementation of high performance filters and other DSP functions. Depending on the device, up to 1,170 I/O user pins can be supported.

The architecture of Stratix II devices is based on two-dimensional row and column that provides signal interconnects between Logic Array Blocks (LAB), memory block structures and DSP blocks. Each LAB consists of eight Adaptive Logic Modules (ALMs), carry chains, LAB control signals, local interconnects and register connection chain lines. An ALM is the Stratix II device family's building block of logic providing efficient implementation of user logic functions. The local interconnect transfers signals between ALMs in the same LAB. Register chain connections transfer the output of an ALM register to the adjacent ALM register in an LAB.

One ALM contains a variety of Look Up Tables (LUT)-based resources that can be divided between two Adaptive LUTs (ALUTs) as shown in Fig. D.9. With up to eight inputs to the combinatorial logic, one ALM can implement various combinations of two functions. In addition to the adaptive LUT-based ressources, each ALM contains two programmable registers, two dedicated full adders, a carry chain, a shared arithmetic chain and a register chain. The ALUT is the cell used in the Quartus II software for logic synthesis. Thus, the number of ALUTs are used throughout in this work as a metric to evaluate the FPGA-based circuit complexity and then the parameter η as defined in [44].

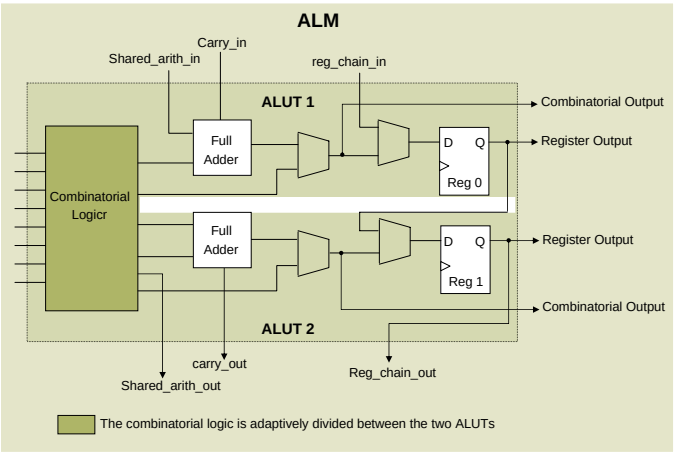

Figure D.9: High-Level block diagram of the Stratix II ALM and its relationship with the ALUTs

Appendix E

LFSR architectures

E.1 Linear feedback shift register basics and its architectures

A linear feedback shift register is a shift register whose input bit is a linear function of its previous state. The only linear functions of single bits are exclusive-OR (XOR) and exclusive-NOR (XNOR); thus it is a shift register whose input bit is driven by the XOR of some bits of the overall shift register value. The value of the inner state s of each register element Z^{-1} could be seen as a recursive linear series, where a_k are the feedback multiplier coefficients:

$$s_{t+n} = a_n s_t + a_{n-1} s_{t+1} + ... + a_1 s_{t+n-1} \tag{E.1}$$

The operation of the register is deterministic and the sequence of values produced by the register is completely determined by its current (or previous) state. Likewise, because the register has a finite number of possible states, it must eventually enter a repeating cycle. However, an LFSR with a well-chosen feedback function can produce a sequence of bits which appears random (pseudo random in nature) and which has a very long cycle.

Two families of LFSR are clearly discriminated in the literature [119]:

- Fibonacci LFSRs and
- Galois LFSRs

These two types are able to carry out dissimilar functionalities and as a consequence, we design two distinct COs, called *Reconfigurable Fibonacci LFSR* as shown in Fig. E.1 and *Reconfigurable Galois LFSR* as shown in Fig. E.2.

E.1.1 RF-LFSR architecture

In the Fibonacci representation as shown in Fig. E.1, the value of each register is given by the linear recurrence (a_k are the feedback multiplier coefficients). The output of Fibonacci LFSR is given by S_0:

$$S_{r-1} = \sum_{1}^{r} a_k S_{r-k} \quad and \quad S_k = S_{k+1}, \quad \forall k \in [0 : r-1] \tag{E.2}$$

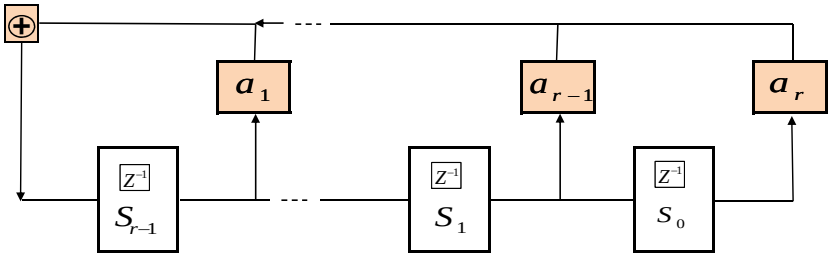

Figure E.1: Fibonacci LFSR

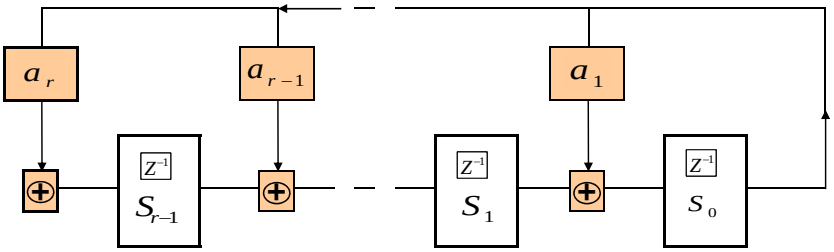

Figure E.2: Galois LFSR

E.1.2 RG-LFSR architecture

In the Galois representation as shown in Fig. E.2 the output of the last cell S_0 is introduced into each of the tapped cells simultaneously, where it is added to the contents of the preceding cell. The value of the current state $S_0^{'}$ of a register n is given by the value of the preceding state S_{i+1} of register $n-1$. The output of Galois LFSR is given by S_0. Taking a_k as the feedback multiplier coefficients, the equation is:

$$S_i^{'} = S_{i+1} + a_{i+1} S_0 \quad \forall k \in [0 : r-1] \tag{E.3}$$

The Galois architecture of LFSRs was designed to operate in a Galois Field (GF). A Galois LFSR can generate all the non zero elements of GF: successive shifts of the register will generate the vector representations of the successive powers of α (the primitive element of GF) [217]. In its simplest form, i.e with *single bit* value, the addition is ensured by XOR and Inverse-XOR.

In practical terms, we first identify an architecture, maximize its executable operations and then define a parameterizable CO. In order to enlarge these workable operations, we undergo the second step of the pragmatic approach and design step by step two NEW CO; the Reconfigurable LFSR (R-LFSR) as in Fig. E.3 and the Extended Reconfigurable LFSR (ER-LFSR) as in Fig. E.4.

E.1.3 R-LFSR architecture

The architecture of the R-LFSR operator is shown in Fig. E.3. This structure is a single bit value operator. In consequence, the parameterizable coefficient is a logical binary AND

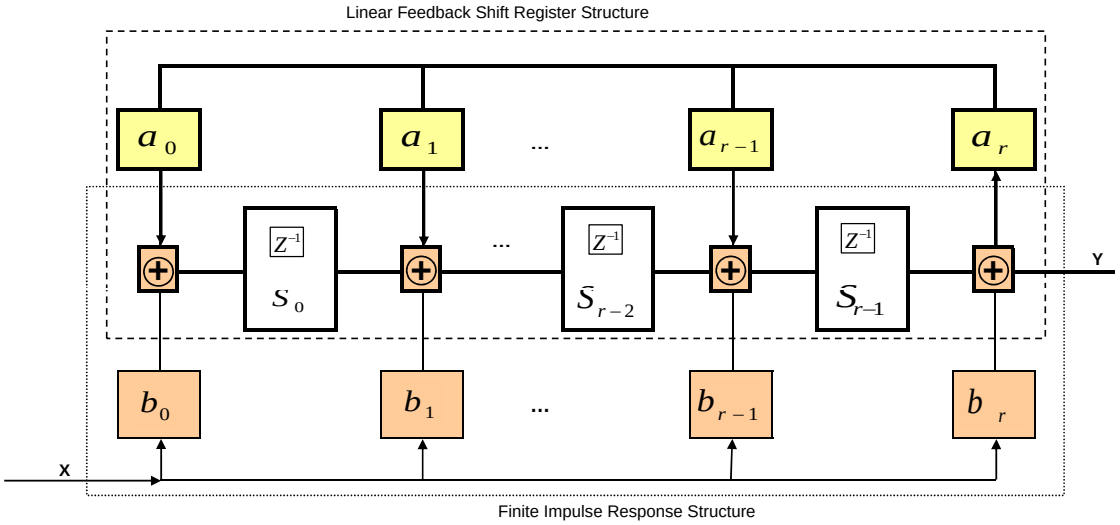

Figure E.3: Architecture of R-LFSR operator

(programmable switch), the adder, a logical binary XOR and the register and a single bit flip flop.
The equation of R-LFSR is:

$$y[n] = \sum_{0}^{N} b_{N-k}.x[n-k] - \sum_{1}^{N} a_{N-k}.y[n-k] \tag{E.4}$$

As highlighted in Fig. E.3 the proposed structure of R-LFSR can be considered as a combination of classical LFSR i.e. Galois model (RG-LFSR) [119, 217] and Finite Impulse Response Filter.

E.1.4 ER-LFSR architecture

The architecture of the ER-LFSR is depicted in Fig. E.4. The proposal structure could be defined by the equations below. The architecture possesses two different outputs and each feedback and feed-forward loop possesses its own parameterizable coefficients.

$$y_G[n] = \sum_{k=0}^{N} b_{N-k}x[n-k] - \sum_{k=1}^{N} a_{N-k}y_G[n-k] + M[n] \tag{E.5}$$

$$M[n] = \sum_{k=1}^{N} a_k^{'}M[n-k] + \sum_{k=1}^{N} a_k^{'}[\sum_{h=0}^{k-1} a_h y_{G[n-(k-h+1)-N]}] + b_h x_{[n-(k-h+1)-N]}] \tag{E.6}$$

$$y_{M1}[n] = \sum_{k=0}^{N} b_k^{'}M[i-k] + \sum_{k=1}^{N} b_k^{'}[\sum_{h=0}^{k} a_h y_{G[n-(k-h)]}] + b_h x_{[n-(k-h]}] + E_i \tag{E.7}$$

In order to itemize the structure and the related equations, the proposed structure could be seen as the combination of four classical Shift Register architectures:

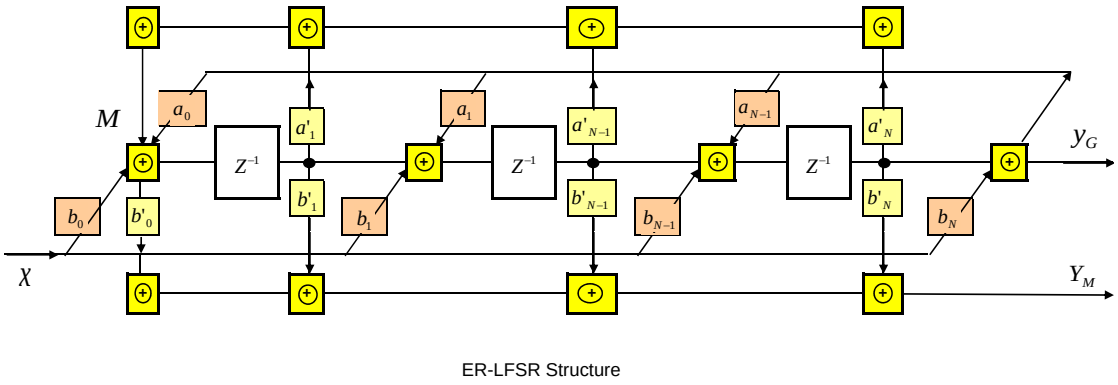

Figure E.4: Architecture of ER-LFSR operator

1. Coefficients a_k specify the linear feedback loop of a Galois LFSR [119, 217].
2. Coefficients a'_k qualify the linear feedback loop of a Fibonacci LFSR [119].
3. Coefficients b_k corresponds to the coefficients a *Transposed form* Finite-Impulse Response Filter.
4. Coefficients b'_k correspond to the coefficients a *Direct form* Finite-Impulse Response Filter.

The combination of architecture 1 and 3 gives the former R-LFSR, already presented in [46] and the combination of 2 and 4 gives the classical structure of an Infinite Impulse Response Filter.

E.2 Preeminent functions of LFSR COs

In this section, we give a brief summary of the operations obtainable by the four COs as in Fig. E.5 (taking the ER-LFSR as a reference). Each LFSR is devoted to a specific *perimeter* defining four *nested* operational perimeters. Since the perimeters are nested, LFSR are called *Matriochkas LFSRs* and define four degree of factoring from the smallest with RF-LFSR to the largest with ER-LFSR. Operations of standards replaced by our LFSRs are listed in Table E.1.

Table E.1: Operations of standards replaced by our LFSRs

Standards	**Function name**	**Polynomial degree**
IEEE 802.11g	Scrambler	7
IEEE 802.11g	Descrambler	7
IEEE 802.11g	CCIT CRC-16 Coder	16
IEEE 802.11g	CCIT CRC-16 Decoder	16
IEEE 802.11g	Convolutional Coder PBCC	2×6
3GPP LTE	CRC-24 Coder	24
3GPP LTE	CRC-16 Coder	16
3GPP LTE	CRC-12 Coder	12
3GPP LTE	CRC-8 Coder	8
3GPP LTE	CRC-24 Decoder	24
3GPP LTE	CRC-16 Decoder	16
3GPP LTE	CRC-12 Decoder	12
3GPP LTE	CRC-8 Decoder	8
3GPP LTE	Convolutional Coder, Rate 1/2	2×8
3GPP LTE	Convolutional Coder, Rate 1/3	3×8
3GPP LTE	Turbo Coder	2×4
IEEE 802.16	Scrambler	15
IEEE 802.16	Scrambler Spreading BPSK	22
IEEE 802.16	Scrambler Pilot Modulation	11
IEEE 802.16	Convolutional Coder	6
IEEE 802.16	CRC-32 Coder	32
IEEE 802.16	CRC-16 Coder	16
IEEE 802.16	CRC-32 Decoder	32
IEEE 802.16	CRC-16 Decoder	16

E.5

	ER-LFSR	R-LFSR	RG-LFSR	RF-LFSR
Generic Structures:				
PN Sequences Generator	X	X	X	X
Scrambler	X	X	X	X
CRC Coder	X	X	X	
CRC Decoder	X	X	X	
Non Systematic Convolutional Coder	X	X		
Recursive Systematic Convolutional Coder	X	X		
Turbo Coder	X	X		
Error Correcting Coder	X	X	X	
Error Correcting Decoder	X	X		
Galois Field Generator	X	X		
Reed Solomon Coder	X	X		
Reed Solomon Decoder	X	X		
"Unconventional" Structures:				
802.11g: Convolutional Coder ERP-PBCC 22/33 Mbits/s	X			
802.16: Turbo Coderr (Rate 1/2 et 1/3)	X			
3 GPP LTE: Scrambler Code Generator (Uplink et Downlink)	X			

	ER-LFSR	R-LFSR	RG-LFSR	RF-LFSR
Binary Operations				
Addition	X	X	X	X
Division	X	X	X	
Multiplication	X	X		
Comparison	X	X		

Figure E.5: Nested functional perimeter of LFSR

List of Figures

List of Tables

Bibliography

[1] F. M. Torre, "Speakeasy-a new direction in tactical communications for the 21st century," in *Proc. Tactical Communications Conference Vol. 1 Tactical Communications: Technology in Transition*, pp. 139–142 vol.1, 1992.

[2] J. H. Reed, *Software Radio: A Modern Approach to Radio Engineering*. Prentice Hall PTR, 2002.

[3] J. P. Cummings, "Software radios for airborne platforms," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 4, pp. 732–747, 1999.

[4] B. Tarver, E. Christensen, A. Miller, and E. R. Wing, "Digital modular radio (DMR) as a maritime/fixed Joint Tactical Radio System (JTRS)," in *Proc. Communications for Network-Centric Operations: Creating the Information Force. IEEE Military Communications Conference MILCOM 2001*, vol. 1, pp. 163–167 vol.1, 2001.

[5] N. Hayes, "The JTRS SCA specification-the past, the present, and the future," in *Proc. IEEE Military Communications Conference MILCOM 2005*, pp. 2713–2719 Vol. 5, 2005.

[6] D. L. Tennenhouse, T. Turletti, and V. G. Bose, "The SpectrumWare testbed for ATM-based software radios," in *Proc.* 5^{th} *IEEE International Conference on Universal Personal Communications Record*, vol. 2, pp. 915–917 vol.2, 1996.

[7] "GNU Radio - The GNU Software Radio," *http://www.gnu.org/software/gnuradio*.

[8] M. Raulet, *Optimisations Mémoire dans la Méthodologie AAA pour Code Embarqué sur Architectures Parallèles*. PhD thesis, Institut National des Sciences Appliquées (INSA) de Rennes, 2006.

[9] C. Moy, M. Raulet, S. Rouxel, J. P. Diguet, G. Gogniat, P. Desfray, N. Bulteau, J. E. Goubard, and Y. Denef, "UML Profiles for Waveform Signal Processing Systems Abstraction," in *SDR Forum Technical Conference*, 2004.

[10] A. Koudri, D. Aulagnier, D. Vojtisek, P. Soulard, C. Moy, J. Champeau, J. Vidal, and S. Lecomte, "RTL Code Generation from MARTE Models, in Proc. of Modeling and Analysis of Real-Time and Embedded Systems with the MARTE UML profile," in *Design, Automation, and Test in Europe (DATE'08)*, 2008.

[11] "SCARI OPEN - Software Communications Architecture - Reference Implementation," *http://www.crc.gc.ca/en/html/crc/home/research/satcom/rars/sdr/products/scari_open/scari_open*.

[12] M. Hermeling, J. Hogg, and F. Bordeleau, "Developing SCA Compliant Systems," *http://www.zeligsoft.com/Technology/Resources.asp*, 2005.

[13] "PrismTech Spectra SDR Power Tools," *http://www.prismtech.com/section-item.asp?snum=3&sid=152*.

[14] W. H. W. Tuttlebee, *Software Defined Radio: Enabling Technologies*. John Wiley & Sons Ltd. UK, 2002.

[15] F. Jondral, *Parameterization; A technique for SDR Implementation*. John Wiley & Sons Inc., London, UK, 2002.

[16] A. R. Rhiemeier, "Benefits and Limits of Parameterized Channel Coding for Software Radio," in *Proc.* 2^{nd} *Karlsruhe Workshop on Software Radios*, pp. 107–112, March 2002.

[17] J. Palicot and C. Roland, "FFT: a Basic Function for a Reconfigurable Receiver," in 10^{th} *International Conference on Telecommunications, ICT'03*, vol. 1, pp. 898–902, March 2003.

[18] L. Alaus, D. Noguet, and J. Palicot, "Extended Reconfigurable Linear FeedBack Shift Register Operators for Software Defined Radio," in *Proc. IEEE* 10^{th} *International Symposium on Spread Spectrum Techniques and Applications ISSSTA '08*, pp. 677–681, 25–28 Aug. 2008.

[19] A. Al Ghouwayel, Y. Louet, and J. Palicot, "A Reconfigurable Butterfly Architecture for Fourier and Fermat transform," in *Proc.* 4^{th} *Karlsruhe Workshop on Software Radios*, pp. 146–151, 17–19 Oct. 2006.

[20] V. Rodriguez, C. Moy, and J. Palicot, "Install or invoke?: The optimal trade-off between performance and cost in the design of multi-standard reconfigurable radios," *Wiley InterScience, Wireless Communications and Mobile Computing Journal*, vol. 7, pp. 1143–1156, November 2007.

[21] J. L. Gross and J. Yellen, *Handbook of Graph Theory (Discrete Mathematics and Its Applications)*. CRC Press, New York, USA., 2004.

[22] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Schmoys, *The Traveling Salesman Problem: A Guided Tour through Combinatorial Optimization*. John Wiley & Sons Inc., 1985.

[23] G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson, "Solution of a large-scale traveling-salesman problem," *Oper. Res.*, vol. 2, pp. 393–410, 1954.

[24] M. W. Padberg and G. Rinaldi, "Optimization of a 532-city symmetric traveling salesman problem by branch and cut," *Oper. Res. Lett.*, vol. 6, pp. 1–7, 1987.

[25] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math*, vol. 1, pp. 269–271, 1959.

[26] J. R. Edmonds, "Paths, trees and flowers," *Canad. J. Math*, vol. 17, pp. 449–467, 1965.

[27] S. A. Cook, "The complexity of theorem-proving procedures," *Proc. 3^{rd} Annual ACM Symp. Theory of Computing, ACM, New York*, pp. 151–158, 1971.

[28] R. M. Karp, *Reducibility among combinatorial problems*. Plenum Press, 1972.

[29] Y. Sorel, "Real-time embedded image processing applications using the A methodology," in *Proc. International Conference on Image Processing*, vol. 1, pp. 145–148, 16–19 Sept. 1996.

[30] M. Raulet, F. Urban, J. F. Nezan, C. Moy, O. Deforges, and S. Y., "SynDEx Executive Kernels for Fast Developments of Applications Over Heterogeneous Architectures," in *European Signal Processing Conference (EUSIPCO'05)*, 2005.

[31] J. P. Delahaye, P. Leray, C. Moy, and J. Palicot, "Managing Dynamic Partial Reconfiguration on Heterogeneous SDR Platforms," in *SDR Forum Technical Conference'05*, 2005.

[32] J. P. Delahaye, J. Palicot, C. Moy, and P. Leray, "Partial Reconfiguration of FPGAs for Dynamical Reconfiguration of a Software Radio Platform," in *IST Mobile and Wireless Communications Summit'07*, 2007.

[33] M. Cummings and S. Haruyama, "FPGA in the software radio," *IEEE Communications Magazine*, vol. 37, pp. 108–112, Feb. 1999.

[34] Lotze, Jorg, Fahmy, A. Suhaib, Noguera, Juanjo, Doyle, Linda, Esser and Robert, "An FPGA-based cognitive radio framework," in *Proc. IET Irish Signals and Systems Conference (ISSC 2008)*, pp. 138–143, 18–19 June 2008.

[35] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic/ Plenum Publishers, New York, 2002.

[36] K. S. R. R. Kondapalli and C. K. Chew, "Optimal Design and Analysis of a DC-DC Synchronous Converter using Genetic Algorithm and Simulated Annealing," *International Journal of Modelling and Simulation*, vol. 3, pp. 4823–4840, 2009.

[37] J. T. Alander, ed., *Proceedings of the Second Nordic Workshop on Genetic Algorithms and their Applications*. Department of Information Technology and Production Economics, University of Vaasa, 1996.

[38] U. B. Nallamottu, T. L. Chambers, and W. Simon, "Comparison of the Genetic Algorithm to Simulated Annealing Algorithm in Solving Transportation Location-allocation Problems With Euclidean Distances," in *Proc. American Society for Engineering Education (ASEE) Gulf-Southwest Annual Conference*, 2002.

[39] L. Ingber, "Adaptive simulated annealing (ASA): Lessons learned," *Journal of Control and Cybernetics: Special issue on Simulated Annealing Applied to Combinatorial Optimization*, 1995.

[40] C. Ortiz-Alemán, R. Martin, J. C. Gamio, and A. Nicolas, "Application of Simulated Annealing and Genetic Algorithms to the Reconstruction of Electrical Permittivity

Images in Capacitance Tomography," in 3^{rd} *World Congress on Industrial Process Tomography*, 2003.

[41] J. Fourier, *Théorie Analytique de la Chaleur/The Analytical Theory of Heat, translated by Alexander Freeman.* Dover Publications, 1822, translated 1878, re-released 2003.

[42] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics Computation*, vol. 19, pp. 297–301, April 1965.

[43] A. Al Ghouwayel, Y. Louet, and J. Palicot, "A reconfigurable architecture for the FFT operator in a software radio context," in *Proc. IEEE International Symposium on Circuits and Systems ISCAS 2006*, p. 4pp., 2006.

[44] W. W. Yu and S. Xing, "Fixed-Point Multiplier Evaluation and Design with FPGA," in *Proc. Society of Photographic Instrumentation Engineers (SPIE)*, 1999.

[45] S. T. Gul, A. Al Ghouwayel, C. Moy, and Y. Louët, "A Novel Design of Reconfigurable Fourier Transform Operator Over C and $GF(F_t)$ for Future Multi-standards SDR Equipments," *International Journal of Communication Networks and Distributed Systems-IJCNDS*, Accepted in June 2009 for publication by the end of 2009.

[46] L. Alaus, D. Noguet, and J. Palicot, "A reconfigurable linear feedback shift register operator for software defined radio terminal," in *Proc.* 3^{rd} *International Symposium on Wireless Pervasive Computing ISWPC 2008*, pp. 319–323, 7–9 May 2008.

[47] S. T. Gul, L. Alaus, C. Moy, J. Palicot, and D. Noguet, "Optimal set of LFSR Common Operators for Multi-Standards Cognitive Radio Terminals," *International Journal of Autonomous and Adaptive Communications-IJAACS, Special Issue on Cognitive Radio Systems*, Accpeted in July 2009 for publication in 2010.

[48] R. Mahesh and A. P. Vinod, "Reconfigurable Frequency Response Masking Filters for Software Radio Channelization," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 55, pp. 274–278, March 2008.

[49] T. Hentschel, "Channelization for software defined base-stations," *Annales des Telecommunications*, vol. 57, pp. 386–420, May-June 2002.

[50] L. Pucker, "Channelization techniques for software defined radio," in *Proc. Spectrum Signal Processing Inc.*, pp. 17–19, November 2003.

[51] C. Kim, S. Y., S. Im, and W. Lee, "SDR-based digital channelizer/de-channelizer for multiple CDMA signals," in *Proc.* 52^{nd} *Vehicular Technology Conference IEEE VTS-Fall VTC 2000*, vol. 6, pp. 2862–2869, 24–28 Sept. 2000.

[52] Y. C. Lim, "Frequency-response masking approach for the synthesis of sharp linear phase digital filters," *IEEE Transactions on Circuits and Systems*, vol. 33, pp. 357–364, Apr 1986.

[53] *IEEE Communications Magazine, Special Issue on Software Radio*, vol. 37. 1999.

[54] J. Mitola III, "The software radio architecture," *IEEE Communications Magazine*, vol. 33, pp. 26–38, May 1995.

[55] M. Woh, S. Seo, H. Lee, Y. Lin, S. Mahlke, T. Mudge, C. Chakrabarti, and F. K., "The Next Generation Challenge for Software Defined Radio," *Springer-Verlag Berlin Heidelberg*, vol. LNCS 4599, pp. 343–354, 2007.

[56] J. Mitola III, *Software Radio Architecture: Object Oriented Approaches to Wireless Systems Engineering.* John Wiley & Sons, Inc., 2000.

[57] J. Mitola III, "Software radios-survey, critical evaluation and future directions," in *Proc. National Telesystems Conference NTC-92*, pp. 13/15–13/23, 19–20 May 1992.

[58] J. Mitola III, "Software radios: Survey, critical evaluation and future directions," *IEEE Aerospace and Electronic Systems Magazine*, vol. 8, pp. 25–36, Apr. 1993.

[59] R. I. Lackey and D. W. Upmal, "Speakeasy: the military software radio," *IEEE Communications Magazine*, vol. 33, no. 5, pp. 56–61, 1995.

[60] R. Baines, "The DSP bottleneck," *IEEE Communications Magazine*, vol. 33, pp. 46–54, May 1995.

[61] *IEEE Journal on Selected Areas in Communications, Special Issue on Software Radio*, vol. 17. 1999.

[62] W. H. W. Tuttlebee, "Software Defined Radio - Baseband Technology for 3G Handsets and Basestations [Book Review]," *Communications Engineer*, vol. 2, pp. 46–47, April–May 2004.

[63] A. Ivers and D. Smith, "A practical approach to the implementation of multiple radio configurations utilizing reconfigurable hardware and software building blocks," in *Proc. MILCOM 97*, vol. 3, pp. 1327–1332, 2–5 Nov. 1997.

[64] A. A. Kountouris, C. Moy, L. Rambaud, and P. Le Corre, "A reconfigurable radio case study: a software based multi-standard transceiver for UMTS, GSM, EDGE and Bluetooth," in *Proc. VTC 2001 Fall Vehicular Technology Conference IEEE VTS 54th*, vol. 2, pp. 1196–1200, 7–11 Oct. 2001.

[65] O. Faust, B. Sputh, D. Nathan, S. Rezgui, A. Weisensee, and A. Allen, "A single-chip supervised partial self-reconfigurable architecture for software defined radio," in *Proc. International Parallel and Distributed Processing Symposium*, p. 7pp., 22–26 April 2003.

[66] H. C. Miranda, P. C. Pinto, and S. B. Silva, "A self-reconfigurable receiver architecture for software radio systems," in *Proc. Radio and Wireless Conference RAWCON '03*, pp. 241–244, 10–13 Aug. 2003.

[67] A. Pacifici, C. Vendetti, F. Frescura, and S. Cacopardi, "A reconfigurable channel codec coprocessor for software radio multimedia applications," in *Proc. International Symposium on Circuits and Systems ISCAS '03*, vol. 2, pp. II–41–II–44, 25–28 May 2003.

[68] T. Hentschel and G. Fettweis, "Sample rate conversion for software radio," *IEEE Communications Magazine*, vol. 38, pp. 142–150, Aug. 2000.

[69] W. A. Abu-Al-Saud and G. L. Stuber, "Efficient sample rate conversion for software radio systems," *IEEE Transactions on Signal Processing*, vol. 54, pp. 932–939, March 2006.

[70] W. A. Abu-Al-Saud and G. L. Stuber, "Modified CIC filter for sample rate conversion in software radio systems," *IEEE Signal Processing Letters*, vol. 10, pp. 152–154, May 2003.

[71] J. A. García, Z. Gulobicic, F. Díaz, J. Alonso, J. Macleod, M. Beach, P. Warr, and D. Jennings, "A TRUST Approach to Software Defined Radio : RF Considerations," tech. rep., TTI (Tecnolgias de Telecomunicaciones y de la Informacion) and University of Bristol., 2001.

[72] M. Mehta, N. Drew, G. Vardoulias, N. Greco, and C. Niedermeier, "Reconfigurable terminals: an overview of architectural solutions," *IEEE Communications Magazine*, vol. 39, pp. 82–89, 2001.

[73] M. Jian, W. H. Yung, and B. Songrong, "An efficient IF architecture for dual-mode GSM/W-CDMA receiver of a software radio," in *Proc. IEEE International Workshop on Mobile Multimedia Communications (MoMuC '99)*, pp. 21–24, 15–17 Nov. 1999.

[74] J. P. Dodley, R. H. Erving, and C. W. Rice, "In-building software radio architecture, design and analysis," in *Proc.* 11^{th} *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications PIMRC-2000*, vol. 1, pp. 479–483, 18–21 Sept. 2000.

[75] W. Schacherbauer, A. Springer, T. Ostertag, C. C. W. Ruppel, and R. Weigel, "A flexible multiband frontend for software radios using high IF and active interference cancellation," in *Proc. IEEE MTT-S International Microwave Symposium Digest*, vol. 2, pp. 1085–1088, 20–25 May 2001.

[76] A. Wiesler, *Parametergesteuertes Software Radio für Mobilfunksysteme.* PhD thesis, Ph.D. dissertation, Forschungsberichte aus dem Institut für Nachrichtentechnik, Universität Karlsruhe (TH), May Karlsruhe, Germany, 2001.

[77] J. Beach, M. MacLeod and P. Warr, *Radio frequency translation for software defined radios in Software Defined Radio: Enabling Technologies.* John Wiley & Sons, London, UK, 2002.

[78] P. B. Kenington and L. Astier, "Power consumption of A/D converters for software radio applications," *IEEE Transactions on Vehicular Technology*, vol. 49, pp. 643–650, March 2000.

[79] J. Singh, "High speed analog-to-digital converter for software radio applications," in *Proc.* 11^{th} *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications PIMRC 2000*, vol. 1, pp. 39–42, 18–21 Sept. 2000.

[80] J. P. Delahaye, J. Palicot, and P. Leray, "A hierarchical modeling approach in software defined radio system design," in *Proc. IEEE Workshop on Signal Processing Systems Design and Implementation*, pp. 42–47, 2–4 Nov. 2005.

[81] G. C. Ahlquist, M. Rice, and B. Nelson, "Error control coding in software radios: an fpga approach," *IEEE Personal Communications*, vol. 6, pp. 35–39, Aug. 1999.

[82] M. C. Valenti, "An efficient software radio implementation of the UMTS turbo codec," in *Proc.* 12^{th} *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 2, pp. G–108–G–113, 30 Sept.–3 Oct. 2001.

[83] V. B. B. M. Thara and M. U. Siddiqi, "Power efficiency of software radio based turbo codec," in *Proc. IEEE Region 10 Conference on Computers, Communications Control and Power Engineering TENCON '02*, vol. 2, pp. 1060–1063, 28–31 Oct. 2002.

[84] A. Wiesler and F. K. Jondral, "A software radio for second- and third-generation mobile systems," *IEEE Transactions on Vehicular Technology*, vol. 51, pp. 738–748, July 2002.

[85] J. Mitola III, "Software radio architecture: a mathematical perspective," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 514–538, April 1999.

[86] R. H. Walden, "Performance trends for analog to digital converters," *IEEE Communications Magazine*, vol. 37, pp. 96–101, Feb. 1999.

[87] H. Tsurumi and Y. Suzuki, "Broadband RF stage architecture for software-defined radio in handheld terminal applications," *IEEE Communications Magazine*, vol. 37, pp. 90–95, Feb. 1999.

[88] C. A. DeVries and R. D. Mason, "Subsampling Architecture for Low Power Receivers," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 55, pp. 304–308, April 2008.

[89] T. Hentschel, M. Henker, and G. Fettweis, "The digital front-end of software radio terminals," *IEEE Personal Communications*, vol. 6, pp. 40–46, Aug. 1999.

[90] E. Newman and D. M. Climek, "Multiband multimode radio (MBMMR) technology applications in the military, civil, and commercial sectors," in *Proc. MILCOM 97*, vol. 3, pp. 1192–1196 vol.3, 1997.

[91] P. G. Cook and W. Bonser, "Architectural overview of the SPEAKeasy system," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 650–661, April 1999.

[92] R. Vidano, "SPEAKeasy II-an IPT approach to software programmable radio development," in *Proc. MILCOM 97*, vol. 3, pp. 1212–1215 vol.3, 1997.

[93] P. A. Eyermann and M. A. Powell, "Maturing the software communications architecture for JTRS," in *Proc. Communications for Network-Centric Operations: Creating the Information Force. IEEE Military Communications Conference MILCOM 2001*, vol. 1, pp. 158–162, 28–31 Oct. 2001.

[94] M. S. Hasan, M. LaMacchia, L. Muzzelo, R. Gunsaulis, L. R. Housewright, and J. Miller, "Designing the Joint Tactical Radio System (JTRS) Handheld, Manpack, and Small form Fit (HMS) Radios for Interoperable Networking and Waveform Applications," in *Proc. IEEE Military Communications Conference MILCOM 2007*, pp. 1–6, 2007.

[95] Chen, Yong, Yuan, Tanya, Le Tourneau and CDR Matt, "Joint Tactical Radio System Common Network Services," in *Proc. IEEE Military Communications Conference MILCOM 2007*, pp. 1–8, 2007.

[96] B. Tarver, E. Christensen, and A. Miller, "The Wireless Information Transfer System (WITS) architecture for the Digital Modular Radio (DMR) software defined radio (SDR)," in *Proc.* 21^{st} *Century Military Communications MILCOM 2000*, vol. 1, pp. 226–230, 22–25 Oct. 2000.

[97] "SDR Forum - Software Defined Radio Forum," *http://www.sdrforum.org*.

[98] "End-to-End Reconfigurability," *www.e2r2.motlabs.com*.

[99] "E^2R at a glance," *ftp://ftp.cordis.europa.eu/pub/ist/docs/directorate_ d/cnt/ e2rii_ en.pdf*.

[100] M. Muck, D. Bourse, K. Moessner, N. Alonistioti, P. Demestichas, E. Nicollet, E. Buracchini, D. Bateman, Z. Boufidis, E. Patouni, V. Stavroulaki, A. Trogolo, and P. Goria, "End-to-End Reconfigurability in Heterogeneous Wireless Systems - Software and Cognitive Radio Solutions enriched by Policy- and Context-based Decision Making," in *Proc.* 16^{th} *IST Mobile and Wireless Communications Summit*, pp. 1–5, 2007.

[101] F. Charot, M. Nyamsi, P. Quinton, and C. Wagner, "Architecture Exploration for 3G Telephony Applications Using aHardware-Software Prototyping Platform," in *Proc. Computer Systems: Architectures, Modeling and Simulation*, 2003.

[102] R. Rabineau, D. Lattard, Y. Durand, L. M., and J. P. Rossi, "Flexible Test-Bed for B3G Systems," in *Proc. IST Mobile and Wireless Communications Summit*, 2006.

[103] V. Bose, *Design and Implementation of Software Radio Using a General Purpose Processor*. PhD thesis, Massachusetts Institute of Technology (MIT), 1999.

[104] A. Hoffmann, H. Meyr, and R. Leupers, *Architecture Exploration for Embedded Processors with LISA*. Kluwer Academic Publishers, 2002.

[105] C. Chavet, P. Coussy, P. Bomel, D. Heller, E. Senn, and E. Martin, *GAUT : a High-Level synthesis tool for DSP applications, In High level Synthesis: from Algorithm to Digital Circuit*. Springer-Verlag, 2006.

[106] S. Rouxel, J. P. Diguet, N. Bulteau, J. Carre-Gourdin, J. E. Goubard, and C. Moy, "UML Framework for PIM and PSM Verification of SDR Systems," in *Proc. SDR Forum Technical Conference*, 2005.

[107] S. Rouxel, G. Gogniat, J. P. Diguet, J. L. Philippe, and C. Moy, *From MDD Concepts to Experiments and Illustrations, In Schedulability Analysis and MDD*. International Scientific and Technical Encyclopedia, 2006.

[108] "PrismTech OpenFusion CORBA Products," *http://www.prismtech.com/section-item.asp?snum=3&sid=251*.

[109] V. Blaschke, F. K. Jondral, S. Nagel, E. Nicollet, and D. Ragot, "Wireless Interoperability for Security - WINTSEC," in *SDR Forum Technical Conference'07*, 2007.

[110] S. Haykin, "Cognitive radio: brain-empowered wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 23, pp. 201–220, Feb 2005.

[111] C. Moy, J. Palicot, V. Rodriguez, and D. Giri, "Optimal Determination of Common Operators for Multi-standards SDR," in *Proc.* 4^{th} *Karlsruhe Workshop on Software Radios*, March 2006.

[112] L. Alaus, J. Palicot, C. Roland, Y. Louët, and D. Noguet, "Promising Technique of Parametrisation For Reconfigurable Radio, the Common Operators Technique: Fundamentals and Examples," *Springer Science + Business Media LLC*, 2009.

[113] F. Jondral, A. Wiesler, and R. Machauer, "A software defined radio structure for 2nd and 3rd generation mobile communications standards," in *Proc.* 6^{th} *IEEE International Symposium on Spread Spectrum Techniques and Applications*, vol. 2, pp. 637–640, 6–8 Sept. 2000.

[114] C. Heegard and S. B. Wicker, eds., *Turbo Coding*. Kluwer Academic Publishers, 1999.

[115] A. Wiesler, H. Schober, R. Machauer, and F. Jondral, "Software radio structure for UMTS and second generation mobile communication systems," in *Proc. VTC 1999 - Fall Vehicular Technology Conference IEEE VTS* 50^{th}, vol. 2, pp. 939–942, Sept. 19–22, 1999.

[116] J. R. Cavallaro and M. Vaya, "Viturbo: a reconfigurable architecture for Viterbi and turbo decoding," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03)*, vol. 2, pp. II–497–500, 6–10 April 2003.

[117] V. Rodriguez, C. Moy, and J. Palicot, "An optimal architecture for a multi-standard reconfigurable radio: Cost-minimising common operators under latency constraints," in 15^{th} *European Information Society Technologies (IST) Summit*, June 2006.

[118] S. T. Gul, C. Moy, and J. Palicot, "Two Scenarios of Flexible Multi-Standard Architecture Designs using a Multi-Granularity Exploration," in *Proc. IEEE* 18^{th} *International Symposium on Personal, Indoor and Mobile Radio Communications PIMRC 2007*, pp. 1–5, 3–7 Sept. 2007.

[119] M. Goresky and A. M. Klapper, "Fibonacci and Galois representations of feedback-with-carry shift registers," *IEEE Transactions on Information Theory*, vol. 48, pp. 2826–2836, Nov. 2002.

[120] E. R. Ferrara, C. F. N. Cowan, and P. M. Grant, *Frequency Domain Adaptive Filtering.* Prentice Hall, 1995.

[121] D. Mansour and J. Gray, A., "Unconstrained frequency-domain adaptive filter," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 30, pp. 726–734, Oct 1982.

[122] T. Schirtzinger, X. Li, and W. K. Jenkins, "A comparison of three algorithms for blind equalization based on the constant modulus error criterion," in *Proc. International Conference on Acoustics, Speech, and Signal Processing ICASSP-95*, vol. 2, pp. 1049–1052, 9–12 May 1995.

[123] K. Berberidis and J. Palicot, "A block quasi-Newton algorithm implemented in the frequency domain," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP-96*, vol. 3, pp. 1731–1734, 7–10 May 1996.

[124] K. Berberidis and J. Palicot, "A frequency-domain decision feedback equalizer for multipath echo cancellation," in *Proc. IEEE Global Telecommunications Conference GLOBECOM '95*, vol. 1, pp. 98–102, 13–17 Nov. 1995.

[125] K. Berberidis, A. Marava, P. Karaivazoglou, and J. Palicot, "Robust and fast converging decision feedback equalizer based on a new adaptive semi-blind channel estimation algorithm," in *Proc. IEEE Global Telecommunications Conference GLOBECOM '01*, vol. 1, pp. 269–273, 25–29 Nov. 2001.

[126] T. Hentschel, G. Fettweis, and M. Bronzel, "Channelization and Sample Rate Adaptation in Software Radio Terminals," in 3^{rd} *ACTS Mobile Communications Summit*, pp. 121–126, June 1998.

[127] E. Bidet, J. M. Cardin, M. D. Kouam, C. Joanblanq, and J. Palicot, "FDF, a 512-TAP FIR filter using a mixed temporal-frequential approach," in *Proc. IEEE Custom Integrated Circuits Conference*, pp. 173–176, 1–4 May 1995.

[128] E. Bidet, D. Castelain, C. Joanblanq, and P. Senn, "A fast single-chip implementation of 8192 complex point FFT," *IEEE Journal of Solid-State Circuits*, vol. 30, pp. 300–305, March 1995.

[129] G. Meigu, "Graphic programming using odd or even points," *Acta Math. Sinica (Chinese Math.)*, vol. 10(1), pp. 263–266/, 1962.

[130] P. Hall, "On representatives of subsets," *J. London Math. Soc.*, vol. 10, pp. 26–30, 1935.

[131] P. R. Halmos and H. E. Vaughan, "The marriage problem," *Amer. J. Math*, vol. 72, pp. 214–215, 1950.

[132] N. Deo, *Graph Theory with Applications to Engineering and Computer Science (Prentice Hall Series in Automatic Computation).* Prentice-Hall, Inc, 1974.

[133] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, pp. 533–547, Sep 1981.

[134] F. Kienle and N. Wehn, "Joint graph-decoder design of IRA codes on scalable architectures [LDPC codes]," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '04)*, vol. 4, pp. iv–673–iv–676, 17–21 May 2004.

[135] O. Boruvka, "O jistém problému minimalnim," *Acta Soc. Sci. Natur. Moravicae*, vol. 3, pp. 37–58, 1926.

[136] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proc. Amer. Math. Soc.*, vol. 7, pp. 48–50, 1956.

[137] R. E. Gomory and T. C. Hu, "Multi-terminal network flows," *SIAM J. Appl. Math.*, vol. 9, pp. 551–556, 1961.

[138] T. C. Hu, "Parallel Sequencing and Assembly Line Problems," *Operations Research*, vol. 9, pp. 841–848, 1961.

[139] M. Garey and D. Johnson, *Computers and Intractablity: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[140] A. R. Rhiemeier and F. K. Jondral, "A software partitioning algorithm for modular software defined radio," in *Proc.* 6^{th} *International Symposium on Wireless Personal Multimedia Communications (WPMC'03)*, pp. 42–46, 2003.

[141] A. R. Rhiemeier and F. K. Jondral, "Mathematical modeling of the software radio design problem," *IEICE Transactions on Communications: Special Issue on Software Defined Radio Technology and Its Applications*, vol. E86-B, pp. 3456–3467, 2003.

[142] "IEEE Standard 802.11b-1999/Cor 1-2001," 2001.

[143] "IEEE Standard 802.11g: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," 2003.

[144] "IEEE Standard 802.16: Air Interface for Fixed Broadband Wireless Access Systems," 2004.

[145] "3GPP Technical Specification TS25.101 (Release 5), User Equipment (UE) radio transmission and reception (FDD), International Telecommunication Union (ITU) Standard," 2005.

[146] "3GPP Technical Specification TS25.102 (Release 5), User Equipment (UE) radio transmission and reception (TDD), International Telecommunication Union (ITU) Standard," 2005.

[147] T. Magnanti and R. Wong, "Network design and transportation planning: models and algorithms," *Transportation Science*, vol. 18, no. 1, pp. 1–56, 1984.

[148] A. Meyerson, K. Munagala, and S. Plotkin, "Cost-distance: Two metric network design," *Foundations of Computer Science. 41st Annual Symposium*, pp. 624–630, 2000.

[149] C. Chekuri, S. Khanna, and J. Naor, "A deterministic algorithm for the cost-distance problem," 12^{th} *Annual ACM-SIAM symposium on discrete algorithms*, pp. 232–233, 2001.

[150] M. V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, R. D. J., and H. H. B., "Bicriteria network design problems," *Journal of Algorithms*, vol. 28, pp. 142–171, 1998.

[151] "IEEE Standard 802.11a: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: high speed physical layer in the 5 GHz band ," 1999.

[152] K. Masselos, S. Blionas, and T. Rautio, "Reconfigurability requirements of wireless communication systems," in *IEEE Workshop on Heterogeneous Reconfigurable Systems on Chip, Chances, Application, Trends*, 2002.

[153] K. Masselos, A. Pelkonen, M. Cupak, and S. Blionas, "Realization of wireless multimedia communication systems on reconfigurable platforms," *Eelsevier, Journal of Systems Architecture*, vol. 49, pp. 155–175, 2003.

[154] Nokia, UNIS and LETI, "Hardware/Software Partitioning," tech. rep., IST and MUMOR, 2003.

[155] "Blackfin 16/32-bit embedded processors," *http://www.analog.com/en/embedded-processing-dsp/blackfin/content/index.html*, 2009.

[156] C. A. Coello, "An updated survey of ga-based multiobjective optimization techniques," *ACM Computing Surveys (CSUR)*, vol. 32, pp. 109–143, 2000.

[157] V. Pareto, *Cours D'Economie Politique*. F. Rouge, Lausanne, Volume I and II, 1896.

[158] G. Syswerda and J. Palmucci, "The Applications of Genetic Algorithms to Resource Scheduling," in *Proc.* 4^{th} *International Conference on Genetic Algorithms*, pp. 502–508, Morgan Kaufmann Publishers, 1991.

[159] P. B. Wilson and M. D. Macleod, "Low implementation cost IIR digital filter design using genetic algorithms," in *IEE/IEEE Workshop on Natural Algorithms in Signal Processing*, (Chelmsford, U. K.), pp. 4/1–4/8, 1993.

[160] W. Jakob, M. Gorges-Schleuter, and C. Blume, "Applications of Genetic Algorithms to Task Planning and Learning," in *Parallel Problem Solving from Nature,* 2^{nd} *Workshop, Lecture Notes in Computer Science*, (Amsterdam), pp. 291–300, North-Holland Publishing Company, 1992.

[161] G. Jones, R. D. Brown, D. E. Clark, P. Willett, and R. C. Glen, "Searching Databases of Two-Dimensional and Three-Dimensional Chemical Structures using Genetic Algorithms," in *Proc.* 5^{th} *International Conference on Genetic Algorithms*, (San Mateo, California, USA), pp. 597–602, Morgan Kaufmann Publishers, 1993.

[162] X. Liu, D. W. Begg, and R. J. Fishwick, "Genetic Approach to Optimal Topology/ Controller Design of Adaptive Structures," *International Journal of Numerical Methods in Engineering*, vol. 41, pp. 815–830, 1998.

[163] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Design*. New York: John Wiley & Sons Inc., 1997.

[164] J. Pearl, *Heuristics*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[165] R. Neapolitan and K. Naimipour, *Foundations of Algorithms*. D.C. Heath and Company, Lexington, Massachusetts, 1996.

[166] G. Brassard and P. Bratley, *Algorithmics: Theory and Practice*. Prentice Hall Englewood Cliffs, 1^{st} Edition, New Jersey, 1988.

[167] P. Husbands, *Genetic Algorithms in Optimization and Adaptation*. Halsted Press, 1992.

[168] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, New Jersey, 1995.

[169] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., 1979.

[170] H. Anton, *Calculus with Analytic Geometryl*. John Wiley & Sons, 2^{nd} Edition, New York, 1984.

[171] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[172] G. B. Lamont, *Compendium of Parallel Programs for the Intel iPSC Computers*. Air Force Institute of Technology, Wright Patterson AFB, 1993.

[173] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *SCIENCE Journal*, vol. 220, pp. 671–680, May 1983.

[174] A. Osyczka, *Multicriterion Optimization in Engineering with FORTRAN Programs*. Ellis Horwood Limited, 1984.

[175] F. Glover and M. Laguna, *Tabu Search*. Kluwer Academic Publishersn, Bostan, Massachusetts, 1997.

[176] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 3rd Edition, 1996.

[177] T. Bäck, *Evolutionay Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

[178] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equations of state calculations by fast computing machines," *Journal of Chemical Physics*, vol. 21, pp. 1087–1092, 1953.

[179] A. Vicini and D. Quagliarella, "Multipoint transonic airfoil design by means of multiobjective genetic algorithms," in 35^{th} *AIAA Aerospace Sciences Meeting and Exhibition*, 1997.

[180] H. P. Schwefel, *Evolution and Optimum Seeking*. John Wiley & Sons, Inc., 1995.

[181] M. Pincus, "A Monte Carlo method for the approximate solution of certain types of constrained optimization problems," *Operations Research*, vol. 18, pp. 1225–1228, 1970.

[182] J. H. Holland, "Adaptation in Natural and Artificial Systems," tech. rep., University of Michigan Press, Ann Arbor, MI, 1975.

[183] B. L. Miller and D. E. Goldberg, "Genetic Algorithms, Tournament Selection, and the Effects of Noise," tech. rep., Department of General Engineering University of Illinois at Urbana-Champaign Urbana, USA, 1995.

[184] S. Reddy and J. Robinson, "Random error and burst correction by iterated codes," *IEEE Transactions on Information Theory*, vol. 18, pp. 182–185, Jan 1972.

[185] S. B. Wicker and V. K. Bhargava, "Reed-Solomon codes and their applications," in *IEEE Press*, New York, 1994.

[186] C. Basile, A. P. Cavallerano, M. S. Deiss, R. Keeler, J. S. Lim, W. C. Luplow, W. H. Paik, E. Petajan, R. Rast, G. Reitmeier, T. R. Smith, and C. Todd, "The US HDTV standard the grand," *IEEE Spectrum*, vol. 32, pp. 36–45, April 1995.

[187] J. M. Pollard, "The fast Fourier transform in a finite field," *IEEE Transactions on Computations*, vol. 25, pp. 365–374, April 1971.

[188] R. C. Agarwal and C. S. Burrus, "Fast digital convolution using Fermat transforms," in *Southwest IEEE Conference Rec., Houston, Texas*, pp. 538–543, April 1973.

[189] C. Rader, "The number theoretic DFT and exact discrete convolution," in *IEEE Arden House Workshop on Digital Signal Processing*, January 1972.

[190] J. Justesen, "On the complexity of decoding Reed-Solomon codes (Corresp.)," *IEEE Transactions on Information Theory*, vol. 22, pp. 237–238, Mar 1976.

[191] C. Y. Fung and S. C. Chan, "A multistage filterbank-based channelizer and its multiplier-less realization," in *Proc. IEEE International Symposium on Circuits and Systems ISCAS 2002*, vol. 3, pp. III–429–III–432, 26–29 May 2002.

[192] W. A. Abu-Al-Saud and G. L. Stuber, "Efficient wideband channelizer for software radio systems using modulated PR filterbanks," *IEEE Transactions on Signal Processing*, vol. 52, pp. 2807–2820, Oct. 2004.

[193] A. Russo, "TPFT-Tunable Pipelined Frequency Transform," in *RF Engines Limited White Paper*, 2002.

[194] A. P. Vinod, E. M. K. Lai, A. B. Premkumar, and C. T. Lau, "A reconfigurable multi-standard channelizer using QMF trees for software radio receivers," in *Proc.* 14^{th} *IEEE on Personal, Indoor and Mobile Radio Communications PIMRC 2003*, vol. 1, pp. 119–123, 7–10 Sept. 2003.

[195] R. Mahesh and A. P. Vinod, "Frequency Response Masking based Reconfigurable Channel Filters for Software Radio Receivers," in *Proc. IEEE International Symposium on Circuits and Systems ISCAS 2007*, pp. 2518–2521, 27–30 May 2007.

[196] Y. C. Lim and Y. Lian, "Frequency-response masking approach for digital filter design: complexity reduction via masking filter factorization," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 41, pp. 518–525, Aug. 1994.

[197] S. T. Gul, C. Moy, and J. Palicot, "Graphical modeling and optimization of air interface standards for Software Defined Radios," in *Proc. IEEE International Multitopic Conference INMIC 2008*, pp. 473–479, 23–24 Dec. 2008.

[198] M. Bellanger, "On computational complexity in digital filters," in *Proc. European Conference on Circuit Theory and Design*, (Hague, Netherlands), pp. 58–63, August 1981.

[199] M. Van Der Schaar, S. Krishnamachari, S. Choi, and X. Xu, "Adaptive cross-layer protection strategies for robust scalable video transmission over 802.11 WLANs," *IEEE Journal on Selected Areas in Communications*, vol. 21, pp. 1752–1763, Dec. 2003.

[200] Y. S. Chan, Y. Pei, Q. Qu, and J. W. Modestino, "On cross-layer adaptivity and optimization for multimedia CDMA mobile wireless networks," in 1^{st} *International Symposium on Control, Communications and Signal Processing*, pp. 579–582, 2004.

[201] Q. Wang and M. A. Abu-Rgheff, "Cross-layer signalling for next-generation wireless systems," in *Proc. IEEE Wireless Communications and Networking WCNC 2003*, vol. 2, pp. 1084–1089, 20–20 March 2003.

[202] W. H. W. Tuttlebee, "Software radio technology: a European perspective," *IEEE Communications Magazine*, vol. 37, pp. 118–123, Feb. 1999.

[203] D. Ikonomou, J. M. Pereira, and J. da Silva, "EU funded R-D on Re-configurable Radio Systems and Networks: The story so Far," in *Infowin Thematic Issue Mobile Communications, ACTS*, 2000.

[204] W. C. Gore, "Transmitting Binary Symbols with Reed-Solomon Codes," in *Proc. Princeton Conference on Information Sciences and Systems*, pp. 495–497, 1973.

[205] A. Michelson, "A Fast Transform in Some Galois Field and an application to Decoding Reed-Solomon Codes," in *IEEE International Symposium on Information Theory*, 1976.

[206] A. Lempel and S. Winograd, "A New Approach of Error Correcting Codes," *IEEE Transactions on Information Theory*, vol. 23, pp. 503–508, 1977.

[207] R. Chien and D. Choy, "Algebraic generalization of BCH-Goppa-Helgert codes," *IEEE Transactions on Information Theory*, vol. 21, pp. 70–79, Jan 1975.

[208] R. E. Blahut, "Transform Decoding without Transform," in 10^{th} *IEEE Communication Theory Workshop, Cypress Gardens, FL*, 1980.

[209] R. E. Blahut, *Algebraic Codes for Data Transmission*. Cambridge University Press, 2003.

[210] W. Peterson, "Encoding and error-correction procedures for the Bose-Chaudhuri codes," *IRE Transactions on Information Theory*, vol. 6, pp. 459–470, September 1960.

[211] D. C. Gorenstein and N. Zierler, "A Class of Error-Correcting Codes in pm Symbols," *Journal of the Society of Industrial and Applied Mathematics*, vol. 9, pp. 207–214, 1961.

[212] R. E. Berlekamp, *Algebraic coding theory*. McGraw-Hill Book Company, Incorporation, New York, 1986.

[213] J. L. Massey, "Shift Register Synthesis and BCH decoding," *IEEE Transactions on Information Theory*, vol. IT-15, pp. 122–127, 1972.

[214] R. T. Chien, "Cyclic Decoding Procedure for the Bose-Chaudhuri-Hocquenghem Codes," *IEEE Transactions on Information Theory*, vol. IT-10, pp. 357–363, October 1964.

[215] G. D. Forney, "On Decoding BCH Codes," *IEEE Transactions on Information Theory*, vol. IT-11, pp. 549–557, 1965.

[216] R. E. Blahut, "A universal Reed-Solomon decoder," *IEEE Transactions on Computations*, vol. C-34, pp. 150–158, March 1984.

[217] P. Kitsos, T. G., and O. Koufopavlou, "An efficient Reconfigurable Multiplier Architecture for Galois Field," *Microelectronics Journal*, 2003.

Publications

International Journal Papers

1. S. T. Gul, Ali Al-Ghouwayel, C. Moy and Y. Louët, "A Novel Design of Reconfigurable Fourier Transform Operator Over C and $GF(F_t)$ for Future Multi-standards SDR Equipments," *International Journal of Communication Networks and Distributed Systems-IJCNDS*, Accepted in June 2009 for publication by the end of 2009.

2. S. T. Gul, L. Alaus, C. Moy, J. Palicot and D. Noguet, "Optimal set of LFSR Common Operators for Multi-Standards Cognitive Radio Terminals," *International Journal of Autonomous and Adaptive Communications-IJAACS, Special Issue on Cognitive Radio Systems*, Accepted in July 2009 for publication in 2010.

3. S. T. Gul, R. Mahesh, C. Moy, J. Palicot and A. P. Vinod "Filter Bank Techniques for SDR Channelizers and their Optimization using Graph Theoretical Approach," *EURASIP Journal on Advances in Signal Processing, Special Issue on Filter Banks for Next Generation Multicarrier Wireless Communications*, Submitted, June, 2009.

International Conference Papers

1. S. T. Gul, C. Moy and J. Palicot, "Graphical Modeling and Optimization of Air Interface Standards for Software Defined Radios," 12^{th} *IEEE International Multitopic Conference-INMIC2008*, Karachi, Pakistan, pp. 473-479, December 2008.

2. S. T. Gul, R. Mahesh, C. Moy, A. P. Vinod and J. Palicot, "A Graphical Approach for the Optimization of SDR Channelizers," *International Union of Radio Science (Union Radio Scientifique Internationale-URSI); XXIX General Assembly*, Chicago, Illinois, USA, August 2008.

3. S. T. Gul, C. Moy and J. Palicot, "Two scenarios of flexible multi-standard architecture designs using a multi-granularity exploration," *The* 18^{th} *Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communication-PIMRC'07*, Athens, Greece, pp. 1-5, September 2007.

International Conference Workshops

1. S. T. Gul, L. Alaus, D. Noguet, C. Moy and J. Palicot "The Common Operator Technique: An Optimization Process to Identify and Design a Set of Common Operators

to Perform SDR Equipment (project NEWCOM++)," *Dynamic Spectrum Management in Cognitive Radio Networks, ICT-MobileSummit 2009*, Santander, Spain, June 2009.

Oral Communications

1. S. T. Gul, "A Multi-granularity Design Exploration for Multi-standard SDR Terminals," *Ecole Supérieure d'Électricité (Supélec), Séminaire SCEE*, Rennes, France, November 2008.

2. S. T. Gul, "Designing Multi-standard SDR Terminals using Common Operators," *Doctoriales de Bretagne*, Brest, France, November 2008.